EQUIPMENT OPERATION MANUAL

Cartesian 200, 300, 400 and 500 Series

Features II

Variables/ Commands/ Functions

Thank you for purchasing the Loctite® Robot.

***Read this manual thoroughly in order to properly use this robot. Be sure to read "For Your Safety" before you use the robot. It will protect you from possible dangers during operation.**

***After having read this manual, keep it in a handy place so that you or the operator can refer to it whenever necessary.**

FOR YOUR SAFETY

Safety Precautions

The precautions stated in this manual are provided for the customer to make the best use of this product safely, and to provide preventive measures against injury to the customer or damage to property.

Be sure to follow the instructions

Various symbols are used in this manual. Please read the following explanations to understand what each symbol stands for.

- **Symbols indicating the Degree of Damage or Danger**

  The following symbols indicate the degree of damage or danger which may be incurred if you neglect the safety notes.

Warnings	These "Warnings" indicate the possibility of death or serious injury.
Cautions	These "Cautions" indicate the possibility of accidental injury or damage to property.

- **Symbols indicating the type of Danger and Preventive Measures**

  The following symbols indicate the type of safety measure that should be taken.

	Indicates the type of safety measure that should be taken.
	Take care. (General caution)
	Indicates prohibition.
	Never do this. (general prohibition)
	Do not disassemble, modify or repair.
	Do not touch. (contact prohibition)
	Indicates necessity
	Be sure to follow instructions.
	Be sure to unplug power supply from wall outlet.
	Be sure to check grounding.

FOR YOUR SAFETY

CARTESIAN Series

⚠ Warnings

Operators who are involved in the programming, inspection and/or maintenance of this robot must take the "special training course" for industrial robots **specified in Article 59 of the Occupational Health and Safety Law and relevant ministry ordinances.**

Do not leave the unit plugged in (power cord and connectors) when it is not in use for long periods of time. Dust can cause fire.
Be sure to shut off the power supply before removing the power cord.

Change the robot's battery periodically (approximately every three years) to prevent malfunction or breakdown.

Keep the emergency stop switch within reach of an operator while teaching and running the robot.
Failure to do so may cause danger since the robot cannot be stopped immediately and safely.

Regularly check that the I/O-S circuits and emergency stop switch work properly.
Failure to do so may cause danger since the robot cannot be stopped immediately and safely.

FOR YOUR SAFETY

⚠ Warnings

Check the mounting screws regularly so that they are always firmly tightened.

Loose screws may cause injury or defect.

Power the unit only with the rated voltage.

Excessive voltage can cause fire or malfunction of the unit.

Do not sprinkle water or oil on the unit, control box, or its cable.

Contact with water can cause electric shock, fire, or malfunction of the unit.
IP Protection Rating is IP40.

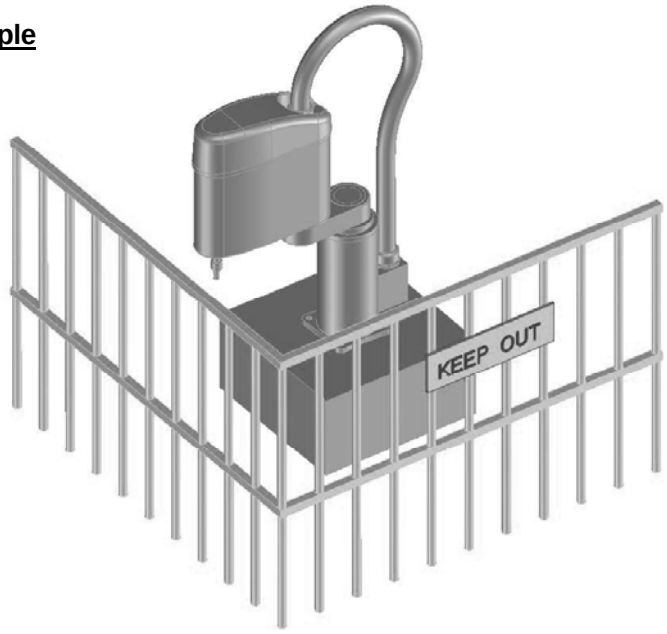
FOR YOUR SAFETY

INSTALLATION

⚠ Warnings

Always use a safety barrier.
A person entering the robot's restricted area may be injured.
At the entry/exit gate of the safety barrier, **install an interlock that triggers an emergency stop when the gate is opened.** Ensure there is no other way of entering the restricted area. Furthermore, **put up a "No Entry" or "No Operating" warning sign** in a clearly visible position.

Example



Install a safety barrier of adequate strength so as to protect the operator from moving tools and flying objects.
Always use protective wear (helmet, protective gloves, protective glasses, and protective footwear) when going inside the safety barrier.

Take adequate precautions against objects the robot is gripping flying or falling off **taking into account the object's size, weight, temperature and chemical composition.**

FOR YOUR SAFETY

⚠ Warnings

Confirm that the unit is properly grounded.
Power supply earth should be connected complying with Type D installation. (under 100 Ω of resistance.)
Insufficient grounding can cause electric shock, fire, or malfunction of the unit.

Plug the power cord into the wall outlet firmly.
Incomplete insertion into the wall outlet makes the plug hot and can cause fire.
Check that the plug is not covered with dust.
Be sure to shut off the power supply before connecting the power cord to the control box.

Place the unit on a suitable flat surface that can support its weight.
Be sure to leave a space greater than 30cm between the back of the robot (equipped with a cooling fan) and the wall.
An insufficient or unstable area can cause the unit to fall, overturn, breakdown, or overheat.

Do not attempt to disassemble or modify the machine.
Disassembly or modification may cause electric shocks, fire or malfunction.

Be sure to use within the voltage range indicated on the unit.
Failure to do so may cause electric shock or fire.

Do not use the unit near inflammable or corrosive gas.
If leaked gas accumulates around the unit, it can cause fire.
IP Protection Rating is IP40.

Place the unit in a well-ventilated area for the health and safety of the operator.

Turn off the unit before inserting and removing cables.
Failure to do so may result in electric shock, fire, or malfunction of the unit.
IP Protection Rating is "IP40."

FOR YOUR SAFETY

⚠ Warnings

Be sure to confirm that all the air tubes are connected correctly and firmly.

Use the robot in an environment between 0 to 40 degrees centigrade with a humidity of 20 to 95 percent without condensation.
Failure to do so may result in malfunction. IP Protection Rating is "IP40."

Use the robot in an environment where no electric noise is present.
Failure to do so may result in malfunction or defect.

Be sure to secure the movable parts of the robot before transportation.
Failure to do so may result in defect or injury.

Do not bump or jar the machine while it is being transported or installed.
This can cause defects.

Use the machine in an environment where it is not exposed to direct sunlight.
Failure to do so may result in malfunction or defect.

Be sure to confirm that tools such as the electric screwdriver unit, etc. are properly connected.
Failure to do so may result in injury or defect.

Be sure to check the wiring to the main unit.
Improper wiring may cause malfunction or defect.

Keep the emergency stop switch within reach of an operator while teaching and running the robot.
Failure to do so may cause danger since the robot cannot be stopped immediately and safely.

Be sure to shut off the power supply before plugging the power cord.

FOR YOUR SAFETY

⚠ Warnings

Be sure to remove the eye bolt after installing the robot so that it does not hit the arm.

Failure to do so may result in injury or breakdown of the unit.

FOR YOUR SAFETY

WORKING ENVIRONMENT

⚠ Warnings

When you lubricate or inspect the unit, unplug the power cord from the control box.
Failure to do so may result in electric shock or injury.
Be sure to shut off the power supply before removing the power cord from the control box.

When going inside the safety barrier, **place a "Do Not Operate" sign** on the start switch.

During operation, always have the emergency stop switch within the operator's reach.
For the operator's safety, the emergency stop switch is necessary to make a quick and safe stop.

Install a safety barrier of adequate strength so as to protect operators from moving tools and flying objects.
Always use protective wear (helmet, gloves, glasses, and footwear) when going inside the safety barrier.

Be sure to confirm that all the air tubes are connected correctly and firmly.

Always watch out for robot's movement, even in the teaching mode.
Special attention will protect the operator from injury.

FOR YOUR SAFETY

DURING OPERATION

⚠ Warnings

When operations are taking place within the safety barrier, **ensure no one enters the robot's restricted area.**

If you must go inside the safety barrier, be certain to **push the emergency stop switch** and **put a "Do Not Operate" sign** on the start switch.

When starting the robot, check that, **no one is within the safety barrier and no object will interfere with the robot operating.**

Under no circumstances should you go inside the safety barrier or place your hands or head inside the safety barrier while the robot is operating.

If anything unusual (e.g. a burning smell) occurs, stop operation and unplug the cable immediately. Contact your dealer or the office listed on the last page of this manual.
Continuous use without repair can cause electric shock, fire, or breakdown of the unit.

During teaching, tests, and actual operation, always have the Emergency stop switch within the operator's reach.
For the operator's safety, the emergency stop switch is necessary to make a quick and safe stop.

FOR YOUR SAFETY

CARTESIAN Series

⚠ Warnings

Be sure to check grounding.
Improper grounding can cause electric shock or fire.

Be sure to use within the voltage range indicated on the unit.
Failure to do so may cause electric shock or fire.

Plug the power cord into the wall outlet firmly.
Failure to do so can cause the input to heat up and may result in fire.
Make sure that the power plug is clean.

Be sure to unplug the power cord from the wall outlet when you examine or grease the machine.
Failure to do so may cause electric shock or fire.

Stop operation and unplug immediately whenever you sense any abnormalities, such as a pungent odor. Immediately contact the dealer from which you purchased the product.
Continued operation may result in electric shock, fire or malfunction.

Install the product in a place which can endure it's weight and conditions while running.
Be sure to leave a space greater than 30cm between the back of the robot (equipped with a cooling fan) and the wall. Installation in an insufficient or unstable place can cause the unit to fall, overturn, breakdown, or overheat.

Be sure to take protective measures such as installing an area sensor or enclosure to avoid injury.
Entering the robot's work range during operation could lead to injury.

Do not attempt to disassemble or modify the machine.
Disassembly or modification may cause electric shocks, fire or malfunction.

FOR YOUR SAFETY

⚠ Warnings

Use the machine indoors where no flammable or corrosive gas is present.
Emission and accumulation of such gasses could lead to fire.
IP Protection Rating is "IP30." ("IP40" for CE specification)

Be sure to unplug the power cord from the wall outlet if the robot will remain unused for long periods of time.
Gathered dust could lead to fire.

Be sure to use power in the proper voltage range.
Failure to do so may result in fire or malfunction.

Keep the unit and the power cables away from water and oil.
Failure to do so may result in electric shock or fire.

Turn off the unit before inserting and removing cables.
Failure to do so may result in electric shock, fire, or malfunction of the unit.
IP Protection Rating is "IP30." ("IP40" for CE specification)

Keep the emergency stop switch within reach of an operator while teaching and running the robot.
Failure to do so may lead to danger since the robot cannot be stopped immediately and safely.

Regularly check that the emergency stop switch works properly.
For models with I/O-S circuits, also check that they work properly.
Failure to do so may lead to danger since the robot cannot be stopped immediately and safely.

FOR YOUR SAFETY

⚠ Cautions

Be sure to check grounding.
Improper grounding may cause malfunction or defect.

Use the Benchtop Robot in an environment between 0 to 40 degrees centigrade with a humidity of 20 to 95 percent without condensation.
Failure to do so may result in malfunction.
IP Protection Rating is "IP30." ("IP40" for CE specification)

Use the machine in an environment where no electric noise is present.
Failure to do so may result in malfunction or defect.

Use the machine in an environment where it is not exposed to direct sunlight. Failure to do so may result in malfunction or defect.

Be sure to confirm that tools such as the electric screwdriver unit, etc. are properly connected.
Failure to do so may result in injury or defect.

Check the mounting screws regularly so that they are always firmly tightened.
Loose screws may cause injury or defect.

Be sure to check the wiring to the main unit.
Improper wiring may cause malfunction or defect.

Be sure to secure the movable parts of the robot before transportation.
Failure to do so may result in defect or injury.

Do not bump or jar the machine while it is being transported or installed.
This can cause defects.

PREFACE

The Loctite® Benchtop Robot CARTESIAN Series is a new low cost, high performance robot. We have succeeded in reducing price while maintaining functionality. Energy and space saving is made possible through the combined use of stepping motors and special micro step driving circuits.

The Loctite® Robot CARTESIAN Series features diverse applications, high speed, rigidity and precision, and can accommodate a wide variety of requirements.

The operation manual consists of the following volumes.

Setup	This volume explains how to set up the robot. * For people who receive safety and installation instructions regarding the robot.
Maintenance	This volume explains how to maintain the robot. * For people who receive safety and installation instructions regarding the robot.
Basic Instructions	This volume provides safety precautions, part names, and the basic knowledge necessary to operate the Benchtop robot.
Dispensing	This volume explains dispensing applications for the Benchtop Series robot.
Quick Start	This volume explains the actual operation of the Benchtop robot with simple running samples.
Teaching Pendant Operation	This volume explains how to operate the robot via the teaching pendant.
PC Operation	This volume explains how to operate the robot from a computer (JR C-Points.)
Features I	This volume explains point teaching.
Features II	This volume explains commands, variables, and functions.
Specifications	This volume provides comprehensive specifications, including mechanical or electrical requirements.

Note) The contents of this volume may be modified without prior notice to improve its quality. Therefore, it may not consistent with the specifications of the delivered series.

Please be sure to follow the instructions described in these volumes. Proper use of the robot will ensure continued functionality and high performance.

The contents described in this volume are based on the standard application. Menu items may vary depending on models.

Be sure to shut off the power supply before plugging in the power cord.

BE SURE TO MAKE A PROPER GROUNDING WHEN YOU INSTALL THE ROBOT.

Be sure to save data whenever it is added or modified. **Otherwise, changes will not be saved if the power to the robot is cut off.**

CONTENTS

Features II

EXPRESSION STRUCTURE

Expression

An expression is fixed numbers, variables, functions (both of String type and Numeric type) and operators combined.

Fixed Number

There are 2 types of fixed numbers, Numeric type (e.g.: 125, 2.0,2e15) and String type (e.g.: "ABC".) String type fixed numbers, characters can be specified in hexadecimal code by using "%." If you want to display "%" on the screen, enter "%%."

e.g.: eoutCOM port2,"%0D%0A" outputs CR LF code.
eoutCOM port2,"%%300" outputs %300.

- If there is any character other than 0 to 9, A to F or % after "%", "%" is dealt with as a character.

Variable

A "variable" is a container into which a value such as numeric values or strings are placed.
Built-in variables, which are built-in as robot features, and user definition variables, which can be freely defined, can be used with this robot.
User definition variables, other than local variables (variables effective only in defined point job data defined by the "declear" command), are defined in the customizing mode. (Refer to the operation manual "Features IV" for a description of the customizing mode.)

Boolean type: 1 bit variable. Keeps a value which is 1 (true)/0 (false) only.
Numeric type: Double type variable
String type: 255 byte variable

Function

Function returns a converted value if values or strings are given.

Both built-in functions, which are built-in as robot features, and user definition functions, which can be freely defined, can be used with this robot.

User definition functions are defined in the customizing mode.

Whether they are Numeric functions or String functions depends on the type of returned values.

Operator

Operator	Description	Value
+	Adds the left and right values.	num
-	Deducts the right value from the left value.	num
*	Multiplys the left and right values.	num
/	Divides the left value by the right value.	num
&	Combines the left and right values. "A" & "B" → "AB"	str
=	Assigns the right value to a left value.	num,str
>	Returns 1 if the left value is larger than the right value. 0 if the left value is smaller or the same.	num,str
<	Returns 1 if the left value is smaller than the right value. 0 if the left value is larger or the same.	num,str
>=,=>	Returns 1 if the left value is larger than the right value or the same. 0 if the left value is smaller.	num,str
<=,=<	Returns 1 if the left value is smaller than the right value or the same. 0 if the left value is larger.	num,str
<>,><	Returns 1 if the left value is not equal to the right value. 0 if they are equal.	num,str
==	Returns 1 if the left value is equal to the right value. 0 if they are not equal.	num,str

<Operator's Priority>

1. Expression in brackets
2. Function, Variable
3. Independent "+", "-"
4. "*", "/"
5. "+", "-", "&"
6. Relational Operator (">", ">=", "=>", "=<", "<=", "<", "<>", "><")
7. Assignment Operator ("=")

COMMAND LIST

If you assign point job data including one of the highlighted ( ) commands to a CP Passing Point, this command will be ignored.

Point Job Commands

Category	Command	Parameters	Content
ON/OFF Output Control	set	Output Destination	ON output
	reset	Output Destination	OFF output
	pulse	Output Destination, Pulse Length	ON pulse output of specified pulse length
	invPulse	Output Destination, Pulse Length	OFF pulse output of specified pulse length
	delaySet	Output Destination, Delay Time	ON output after specified delay time
	delayReset	Output Destination, Delay Time	OFF output after specified delay time
	onoffBZ	ON Time, OFF Time	Sounds an alarm buzzer off and on.
	onoffGLED	ON Time, OFF Time	The LED (Green) on the front body blinks. (for CARTESIAN only)
	onoffRLED	ON Time, OFF Time	The LED (Red) on the front body blinks. (for CARTESIAN only)
	dataOut	Output Value, Output Destination, Pulse Length	Outputs a tag code assigned to numeric data or a point to I/O.
	dataOutBCD	Output Value, Output Destination, Pulse Length	Outputs numeric data or a tag code assigned to a point to I/O in BCD.
if Branch, Wait Condition	if	-	if Branch
	then	-	Execute if true.
	else	-	Execute if false.
	endIf	-	End of if Branch
	waitCondTime	Period for Time Out	Wait for conditions for a certain period
	timeUp	-	Execute when time is up.
	endWait	-	End of WAIT command
	waitCond	-	Wait conditions.

Category	Command	Parameters	Content
Condition	ld	Boolean Variable or Expression	ON input
	ldi	Boolean Variable or Expression	OFF input
	and	Boolean Variable or Expression	Serial ON input
	ani	Boolean Variable or Expression	Serial OFF input
	or	Boolean Variable or Expression	Parallel ON input
	ori	Boolean Variable or Expression	Parallel OFF input
	anb	-	Blocks serial connection
	orb	-	Blocks parallel connection
Delay	delay	Wait Time	Stops for the specified time.
	dataIn	Read Out Source, Read Out Width	Reads out numeric data from I/O.
	dataInBCD	Read Out Source, Read Out Width	Reads numeric data in BCD from I/O.
	waitStart	-	Waits for a start signal.
	waitStartBZ	-	Waits for a start signal while acknowledging an error with an alarm buzzer.
Pallet	loopPalette	Pallet Number, Destination Number	Repetition loop of Pallet
	resPalette	Pallet Number	Resets pallet counter.
	incPalette	Pallet Number	Increases pallet counter number. (+1)
Execute Flow Control	callBase	-	Calls a point job defined by type at the user definition type point to which a Point Job Number is set.
	callJob	Point Job Number	Calls a subroutine of point job data specified by number.
	callPoints	Point String Identifier	Executes a specified point string (defined in Customising mode).
	returnJob	-	Point job end
	returnFunc	Expression	Assigns a value of the specified expression as a returned value and ends the function. (Valid in functions only)
	callProg	Program Number	Calls a subroutine of a program specified by number.
	endProg	-	Program end.
	goPoint	Drive Condition Number, Pont Number	Jumps to a specified point.
	goRPoint	Drive Condition Number, Relative Point Number	Jumps to a point specified relatively.
	goCRPoint	Drive Condition Number, Destination Point Selection	Jumps to a selected destination point during CP drive.

Category	Command	Parameters	Content
for, do-loop	for	Control Variable, Initial Value, End Value, Step Value	Repeats commands from “for” to “next” until the specified variable changes from Initial Value to End value.
	next	-	
	exitFor	-	Exits “for” sentence.
	do	-	Repeats commands from “do” to “loop.”
	loop	-	
	exitDo	-	Exits from “do” sentence.
Move	upZ	Distance, Speed	Up Z Axis.
	donwZ	Distance, Speed	Down Z Axis.
	movetoZ	Distance, Speed	Move Z Axis.
	lineMoveSpeed	Speed, Distance or Roatation	Shifts the specified distance (relative distance) at the specified speed in CP line drive. (Relative move command)
	lineMoveX	X Distance	Relative move command in the X direction
	lineMoveY	Y Distance	Relative move command in the Y direction
	lineMoveZ	Z Distance	Relative move command in the Z direction
	lineMoveR	R Rotation	Relative move command in the R direction
	initMec	Axis	Mechanical initialization of a specified axis (for CARTESIAN only)
	checkPos	-	Position error detection (for CARTESIAN only)
LCD, 7SLED	clrLCD	-	Clears the LCD display.
	clrLineLCD	Rows (1-13)	Clears a specified line on the LCD display.
	outLCD	Rows (1-13), Columns (1-40), Character String	Displays character strings on the LCD display.
	eoutLCD	Rows (1-13), Columns (1-40), Character String Expression	Displays the result of a character string expression on the LCD display.
	sys7SLED	-	Returns the display of the 7 segment LED changed by ”out7SLED” to the previous program number. (for CARTESIAN only)
	out7SLED	Display Type, Display Value	7 segment LED output (for CARTESIAN only)
	clrCOM	Port	Clears a buffer received from the COM.
	shiftCOM	Port, Shift Number	Shifts data received from the COM. Deletes data from the top to the Shift Number.
	stopPC	-	Stops PC communication of COM1.
	startPC	-	Starts PC communication of COM1.

Category	Command	Parameters	Content
COM Input/Output	outCOM	Port, Character String	Outputs a character string from the COM.
	eoutCOM	Port, Character String Expression	Outputs the result of an expression from the COM.
	setWTCOM	Port, Wait Time	Sets Wait Time (Period for Time Out) for receiving from the COM.
	InCom	Variable Name, Port, Wait Time	Assigns data received from the COM to the specified variable.
	cmpCOM	Port, Character String	Compares received data with a character string. The result is entered into System Flag (sysFlag(1) to (20).)
	ecmpCOM	Port, Character String Expression	Compares received data with a character string expression. The result is entered into System Flag (sysFlag(1) to (20).)
	clrCOM	Port	Clears a buffer received from the COM.
	shiftCOM	Port, Shift Number	Shifts data received from the COM. Deletes data from the top to the Shift Number.
	stopPC	-	Stops PC communication of COM1.
	startPC	-	Starts PC communication of COM1.
Variable, Comment, System Control	declear	Type, Identifier	Declare a local variable.
	let	Character String of a assignment expression	Assigns data calculated in the left-hand expression to a variable in the right-hand expression. +, -, *, /, =, (,), & can be used.
	rem	Character String	One line comment
	crem	Character String	Comment at the end of a command line
	setProgNo	Program Number	Changes the Program number. * Do not execute this command while running. If you want to run another program while running, use the command “callProg.”
	setSeqNo	Sequencer Number	Changes the Sequencer program number of “system data.”
Z Sensor	takeZWadj	Work Adjustment Number	Calculates the adjusting amount in the Z direction with data gained from the Distance sensor/Touch sensor using the [Workpiece Adjustment] setting.

Execute Condition

Category	Command	Parameters	Content
Condition	ld	Boolean Variable or Expression	ON input
	ldi	Boolean Variable or Expression	OFF input
	and	Boolean Variable or Expression	Serial ON input
	ani	Boolean Variable or Expression	Serial OFF input
	or	Boolean Variable or Expression	Parallel ON input
	ori	Boolean Variable or Expression	Parallel OFF input
	anb	-	Blocks serial connection
	orb	-	Blocks parallel connection

Sequencer

Category	Command	Parameters	Content
Condition	ld	Boolean Variable or Expression	ON input
	ldi	Boolean Variable or Expression	OFF input
	and	Boolean Variable or Expression	Serial ON input
	ani	Boolean Variable or Expression	Serial OFF input
	or	Boolean Variable or Expression	Parallel ON input
	ori	Boolean Variable or Expression	Parallel OFF input
Coil	out	Output Destination	Coil drive
	set	Output Destination	Movement Holding Set
	reset	Output Destination	Movement Holding Reset
	pls	Output Destination	Output of the Leading Edge of Pulse
	plf	Output Destination	Output of the Trailing Edge of Pulse
Cnnection	anb		Parallel Connection of Serial Circuit Blocks
	orb		Serial Connection of Parallel Circuit Blocks
	mps		Stores data in process of calculation
	mrd		Reads out data in process of calculation
	mpp		Reads out and resetting data in process of calculation
Other	nop		No process

VARIABLE LIST

Built-in variables, which are built-in as robot functions, and user definition variables, which can be freely defined, can be used with this robot.
User definition variables, except local variables (variables effective only in defined point job data which are defined by the “declear” command), are defined in the customizing mode. (Refer to the operation manual “Features IV” for a description of the customizing mode.)

Boolean type 1 bit variable (Holds only a value of 1 (true) or 0 (false).)
Numeric type 8 bytes real type (double type) variable.
String type 255 bytes variable

Category	Type	Identifier	Description
Free Variable	boo	#mv(1 to 99)	Boolean variable
	boo	#mkv(1 to 99)	Boolean variable (keeping variable)
	num	#nv(1 to 99)	Numeric variable
	num	#nkv(1 to 99)	Numeric variable (keeping variable)
	str	#sv(1 to 99)	String variable
	str	#skv(1 to 99)	String variable (keeping variable)
Input Variable	boo	#sysIn1 to 15	I/O-SYS
	boo	#genIn1 to 18	I/O-1
	boo	#handIn1 to 4	I/O-H
Output Variable	boo	#sysOut1 to 14	I/O-SYS
	boo	#genOut1 to 22	I/O-1
	boo	#handOut1 to 4	I/O-H
System Flag	boo	#sysFlag(1) to #sysFlag(999)	Refer to the system flag table.
Buzzer	boo	#FBZ	set #FBZ : Sound buzzer. reset #FBZ : Stop buzzer. (onoffBZ : Sound buzzer intermittently.)
Special Variable	num	#downTimer1 to 10	When a value is assigned, it counts down automatically. (msec units)
	num	#jobStartHight	Start a point job from an assigned value above the set point Z coordinate. (Disabled in CP drive)
Pallet	boo	#palletFlag(1 to 100)	Pallet flag (corresponds to pallet 1 to 100)
	num	#palletCount(1 to 100)	Pallet counter (corresponds to pallet 1 to 100)
Workpiece Adjustment	num	#workAdj_X(1 to 100) #workAdj_Y(1 to 100) #workAdj_Z(1 to 100) #workAdj_R(1 to 100) #workAdj_Rotation(1 to 100)	Workpiece adjustment amount of each axis (Corresponds to workpiece adjustment 1 to 100)

- “Keeping variable” is a variable which holds its value even when the robot power is turned off.

[Variables]

Category	Type	Identifier	Description
Sequencer program	boo	#seqT(1 to 99)	Set to 1 when #seqTCount reaches the given value or greater.
	num	#seqTCount (1 to 50): Integrating Timer #seqTCount (51 to 99): Unintegrating Timer	One counter can count from 0.001 to 2,147,483,647 seconds. (0.001 sec increment)
	boo	#seqC (1 to 99)	1 when #seqCCount reaches the given value or greater.
	num	#seqCCount (1 to 99)	One counter can count from 1 to 2,147,483,657.
Current point coordinates	num	#point_X	X coordinate value of current point
	num	#point_Y	Y coordinate value of current point
	num	#point_Z	Z coordinate value of current point
	num	#point_R	R coordinate value of current point
	num	#point_TagCode	Tag code value of current point
Given point coordinates	num	#P_X(1 to last point count)	X coordinate value of given point
	num	#P_Y(1 to last point count)	Y coordinate value of given point
	num	#P_Z(1 to last point count)	Z coordinate value of given point
	num	#P_R(1 to last point count)	R coordinate value of given point
	num	#P_TagCode(1 to last point count)	Tag code value of given point
Given program Given point coordinates	num	#prog_P_X (1 to 255, 1 to last point count)	X coordinate value of given program and given point.
	num	#prog_P_Y (1 to 255, 1 to last point count)	Y coordinate value of given program and given point.
	num	#prog_P_Z (1 to 255, 1 to last point count)	Z coordinate value of given program and given point.
	num	#prog_P_R (1 to 255, 1 to last point count)	R coordinate value of given program and given point.
	num	#prog_P_TagCode (1 to 255, 1 to last point count)	Tag code value of given program and given point

FUNCTION LIST

Built-in functions, which are built in as robot functions, and user definition functions, which can be freely defined, can be used with this robot.
User definition functions are defined in the customizing mode. (Refer to the operation manual "Features IV" for a description of the customizing mode.)

x, y: Numeric value or numeric variable
n, m: Numeric value made a certain digit or greater by rounding or truncation
a, b: String or string variable

Category	Type	Identifier	Description
Robot system	num	currentMainProgNumber()	Main program No. currently running
	num	currentSubProgNumber()	Subprogram No. currently running
	num	currentPointNumber()	Point No. currently running
	num	currentArmX()	Current arm coordinate X, [mm] units
	num	currentArmY()	Current arm coordinate Y, [mm] units
	num	currentArmZ()	Current arm coordinate Z, [mm] units
	num	currentArmR()	Current arm coordinate R, [degree] units
	num	NumCOM (COM Port Number)	Data byte count of COM receiving port
Arithmetic system	num	abs(x)	Absolute value
	num	max(x,y)	Maximum value
	num	min(x,y)	Minimum value
	num	degrad(x)	Conversion from degree to radian (x* π/180)
	num	raddeg(x)	Conversion from radian to degree (x*180/π)
	num	sqrt(x)	Square root
	num	sin(x)	Sine
	num	cos(x)	Cosine
	num	tan(x)	Tangent
	num	atan(x)	Arc tangent
	num	atan2(x,y)	Arc tangent
	num	int(x)	Large integer that does not exceed x. Ex: int (1.3)=1, int (-1.3)=-2
	num	ip(x)	Integer part of x. sgn (x)*int (abs(x)) (When x is a negative number, sgn (x) becomes -1 and when x is a positive number, sgn (x) becomes +1.) Ex: ip (1.3)=1, ip (-1.3)=-1
	num	fp(x)	Decimal part of x x-ip (x) Ex: fp (1.3)=0.3, fp (-1.3)=-0.3
	num	mod(x,y)	Value of x which makes y modulo x-y*int (x/y)
	num	remainder(x,y)	Remainder of dividing x by y x-y*ip (x/y)
	num	pow(x,y)	x to the power of y

x, y: Numeric value or numeric variable

n, m: Numeric value made a certain digit or greater by rounding or truncation

a, b: String or string variable

Category	Type	Identifier	Description
String system	str	chr(x)	Returns a string (1 character) with the given character code.
	num	ord(a)	Returns the top string code. Other codes are ignored.
	num	len(a)	Returns the string length (byte length). Does not handle multiple byte codes.
	num	strPos(a,b)	Returns the first part string position matching b in a.
	str	strMid(a,n,m)	Returns a character string from the n th to the m th characters, counting from the top of character string a.
	str	str(x)	Converts a numeric value to a decimal string.
	str	strBin(n,m)	Converts a numeric value to a binary string. m: Number of binary string digits
	str	strHex(n,m)	Converts a numeric value to a hexadecimal string. m: Number of hexadecimal string digits
	str	str1SI(x)	Rounds a numeric value to a 1 byte signed integer, and converts it to a 1 byte string. (1 byte Signed Integer)
	str	str2SIBE(x)	Rounds a numeric value to a 2 byte signed integer and converts it to a 2 byte string big endian. (2 byte Signed Integer Big Endian)
	str	str2SILE(x)	Rounds a numeric value to a 2 byte signed integer and converts it to a 2 byte string little endian. (2 byte Signed Integer Little Endian)
	str	str4SIBE(x)	Rounds a numeric value to a 4 byte signed integer and converts it to a 4 byte string big endian. (4 byte Signed Integer Big Endian)
	str	str4SILE(x)	Rounds a numeric value to a 4 byte signed integer and converts it to a 4 byte string little endian. (4 byte Signed Integer Little Endian)
	str	str4FBE(x)	Regards a numeric value as a floating decimal and converts it to 4 byte string big endian. (4 byte Float Big Endian)
	str	str4FLE(x)	Regards a numeric value as a floating decimal and converts it to 4 byte string little endian. (4 byte Float Little Endian)
	str	str8DBE(x)	Regards a numeric value as a double floating decimal and converts it to 8 byte string big endian. (8 byte Double Big Endian)
	str	str8DLE(x)	Regards a numeric value as a double floating decimal and converts it to 8 byte string little endian. (8 byte Double Little Endian)
	num	val(a)	Regards a string as a decimal string and converts it to a numeric value.
	num	valBin(a)	Regards a string as a binary string (list of “0” and “1”) and converts it to a numeric value.
	num	valHex(a)	Regards a string as a hexadecimal string (list of “0” to “1”, “A” to “F”,, or “a” to “f”) and converts it to a numeric value.
	num	val1SI(a)	Regards the top character as a 1 byte signed integer and converts it. (1 byte Signed Integer)
	num	val2SIBE(a)	Regards the top 2 characters as a 2 byte signed integer big endian and converts it. (2 byte Signed Integer Big Endian)

x, y: Numeric value or numeric variable

n, m: Numeric value made a certain digit or greater by rounding or truncation

a, b: String or string variable

Category	Type	Identifier	Description
String system	num	val2SILE(a)	Regards the top 2 characters as a 2 byte signed integer little endian and converts it. (2 byte Signed Integer Little Endian)
	num	val4SIBE(a)	Regards the top 4 characters as a 4 byte signed integer big endian and converts it. (4 byte Signed Integer Big Endian)
	num	val4SILE(a)	Regards the top 4 characters as a 4 byte signed integer little endian and converts it. (4 byte Signed Integer Little Endian)
	num	val4FBE(a)	Regards the top 4 characters as a floating decimal big endian and converts it. (4 byte Float Big Endian)
	num	val4FLE(a)	Regards the top 4 characters as a floating decimal little endian and converts it. (4 byte Float Little Endian)
	num	val8DBE(a)	Regards the top 8 characters as a double floating decimal big endian and converts it. (8 byte Double Big Endian)
	num	val8DLE(a)	Regards the top 8 characters as a double floating decimal little endian and converts it. (8 byte Double Little Endian)
	num	valSum(a)	Returns the sum of a string code added from top to bottom.
	num	valCRC(a)	Remainder of dividing a character string as a bit string by generation polynomial $X^{16}+X^{12}+X^{5}+1$
	str	bitNot(a)	Bit inversion
	str	bitAnd(a,b)	Bit logical multiple
	str	bitOr(a,b)	Bit logical add
	str	bitXor(a,b)	Bit exclusive logical add

[Variables]

Free variables: #mv, #mkv, #nv, #nkv, #sv, #skv

A “variable” is a container into which a value is placed. This robot has the following variables which can be used freely. When using the following variables, variable declaration is unnecessary.

	Identifier	
Free variables	#mv(1 to 99)	Boolean variable
	#mkv(1 to 99)	Boolean variable (keeping variable)
	#nv(1 to 99)	Numeric variable
	#nkv(1 to 99)	Numeric variable (keeping variable)
	#sv(1 to 99)	String variable
	#skv(1 to 99)	String variable (keeping variable)

- “Keeping variable” is a variable that retains its value even when the robot power is turned off.

#mv(1 to 99), #mkv(1 to 99)
Boolean variable. “Boolean variable” is a variable that can hold a 1-bit value of 0 or 1. It can be used as a condition operation expression (ld, ldi) or assignment expression (let) parameter.

- A sequencer program can also use Boolean type free variables (#mv (1 to 99), #mkv (1 to 99)).

#nv (1 to 99),#nkv (1 to 99)
Double type numeric variable that can be used as an assignment expression (let) parameter.

#sv (1 to 99),#skv (1 to 99)
String variable that can hold up to 255 bytes. When used as an assignment expression (let) parameter, assignment by “=” and connection by “&” are possible.

Input variables:#sysIn1 to 15, #genIn1 to 18, #handIn1 to 4

An input variable is a Boolean variable that can be referenced only. A value cannot be written.
The input variables correspond to the I/O-SYS, I/O-1, and I/O-H input pins. When an ON signal is received, the input variable becomes “1” (true).

Category	Identifier (JS Series)	Identifier (Cartesian)	Connector	Description
Input variable	#sysIn1 to 15	#sysIn1 to 16	I/O-SYS	Reference only Boolean variable
	#genIn1 to 18	#genIn1 to 8	I/O-1	Reference only Boolean variable
	#handIn1 to 4		I/O-H	Reference only Boolean variable

#sysIn1 to 16 (I/O-SYS) are assigned a function in advance.
e.g.)#sysIn1: Start signal (When ON signal received, operation starts.)

If you want to use #sysIn1 to 16 (I/O-SYS) for a function other than the one to which it has been assigned, switch the run mode parameter setting to free. (I/O-SYS function assignment)

Output variables:

#sysOut1 to 15, #genOut1 to 18, #handOut1 to 4

Output variables are Boolean variables.
Output variables corresponds to the I/O-SYS, I/O-1, and I/O-H output pins. When an ON signal is output, the output variables become “1” (true).

Category	Identifier (JS Series)	Identifier (Cartesian)	Connector	Description
Output variable	#sysOut1 to 14	#sysOut1 to 16	I/O-SYS	Boolean variable
	#genOut1 to 18	#genOut1 to 8	I/O-1	Boolean variable
	#handOut1 to 4		I/O-H	Boolean variable

#sysOut1 to 16 (I/O-SYS) are assigned a function in advance.
Ex) #sysOut1: Ready for Start (ON signal: Operation can be started)

If you want to use #sysIn1 to 16 (I/O-SYS) for a function other than the one to which it has been assigned, switch the run mode parameter setting to free. (I/O-SYS function assignment)

Down timer : #downTimer1 to 10

Numeric variable. When assigned (let) a value, counting down in msec units starts automatically. A value can also be assigned during countdown.
The maximum value that can be assigned is 2,147,483,647 (msec).

Category	Identifier	Description
Special variable	#downTimer1 to 10	When a value is assigned, counting down (in msec units) starts automatically.

Point job starting height : #jobStartHight

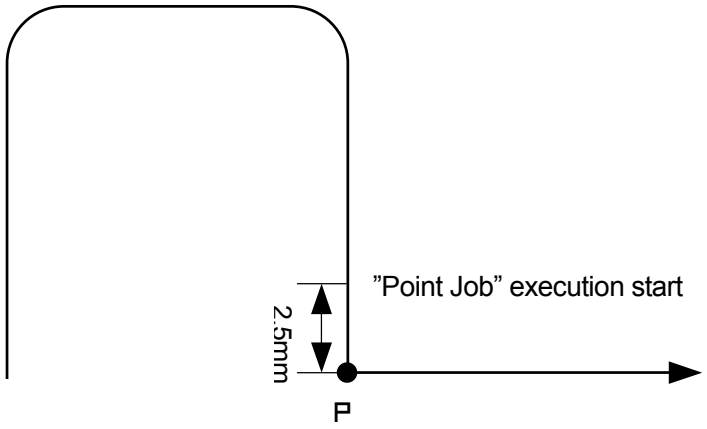
When a value is assigned (let) to the variable “#jobStartHight” before movement or during movement, the point job starts from an assigned value above the set point Z coordinate.
Setting point job data that includes “#jobStartHight” at “point job” is meaningless because the robot has already arrived at the point job start position. Also, since this variable acts only on the set point, the point job start position of the next point does not change.

e.g.)

P01
Type: CP start point
Job before moving: Point job data 3
Point job: Point job data 12

Point Job Data 3

#jobStartHight 25



Category	Identifier	Description
Special variable	#jobStartHight	Point job starts from a given value above the Z coordinate of the point. (Invalid in CP drive)

Pallet : #palletflag (1 to 100), #palletCount (1 to 100)

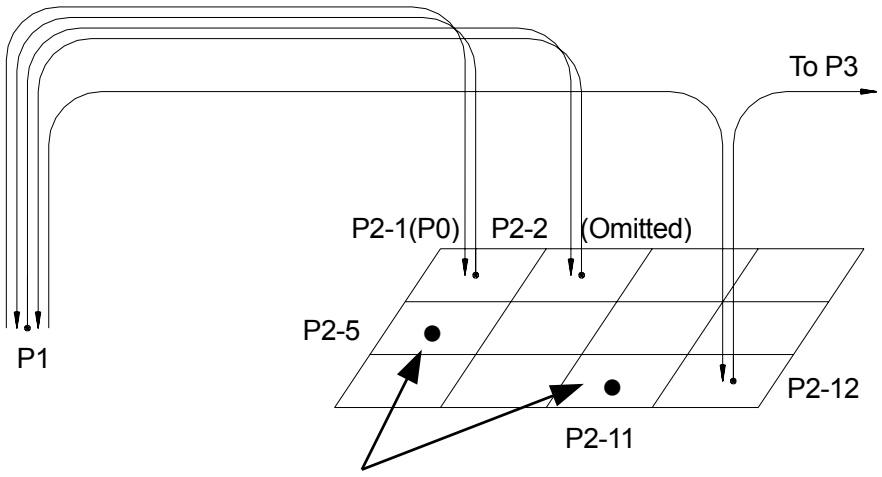
#palletCount(1 to 100) is a numeric variable and #palletflag(1 to 100) is a Boolean variable.
These variables retain the value of the pallet counter and pallet flag of the “Pallet Routine” under Additional function data (1 (true) when the pallet counter is full).
By using this variable, a pallet can be moved to the next point, etc. midway or a given pallet can be skipped.

Category	Identifier	Description
Pallet	#palletflag (1 to 100)	Pallet flag (corresponds to pallet 1 to 100)
	#palletCount (1 to 100)	Pallet counter (corresponds to pallet 1 to 100)

- #palletFlag (1 to 100) does not become “1” (true) even if a value which fills the counter is assigned to #palletcount (1 to 100).

e.g.) Perform a pallet job by skipping a given pallet.
Pick up a workpiece at P1, places it on the pallet (set at P2) and advances to the next point (P3) where the pallet becomes full. However, there are 2 points (P2-5, P2-11) on the pallet where a workpiece is not placed.

The pallet becomes pallet number 3, and the tool is connected using the following settings.
“Pick up workpiece” (pick up): #handOut1=ON
”Place workpiece” (place): #handOut1=OFF



Points where work is not performed

Point job data set on P1

set #handOut1	Pick up workpiece.

Point job data set on P2

if
ld #palletCount(3) = 5
or #palletCount(3) = 11
else
reset #handOut1
endIf
loopPalette 3,1
	If
	#palletCount (3) is other than 5 (P2-5), 11 (P2-11),
	
	Place (release) workpiece.
	
	Increase the pallet 3 counter by 1. If the counter becomes full, advance to the next command.
	(In this case, the point job ends because there is no next command.)
	If the counter is not full, move to P1.

Workpiece adjustment: #workAdj_X, #workAdj_Y, #workAdj_Z, #workAdj_R, #workAdj_Rotation

Numeric variables. These variables hold the adjustment amount and rotation adjustment amount of each axis of the [Workpiece Adjustment] under Additional function data.

Category	Identifier	Description
Workpiece adjustment	#workAdj_X (1 to 100)	Workpiece adjustment amount in the X direction (Corresponds to workpiece adjustment 1 to 100)
	#workAdj_Y (1 to 100)	Workpiece adjustment amount in the Y direction (Corresponds to workpiece adjustment 1 to 100)
	#workAdj_Z (1 to 100)	Workpiece adjustment amount in the Z direction (Corresponds to workpiece adjustment 1 to 100)
	#workAdj_R (1 to 100)	Workpiece adjustment amount in the R direction (Corresponds to workpiece adjustment 1 to 100)
	#workAdj_Rotation (1 to 100)	Workpiece adjustment amount by rotating angle (Corresponds to workpiece adjustment 1=100)

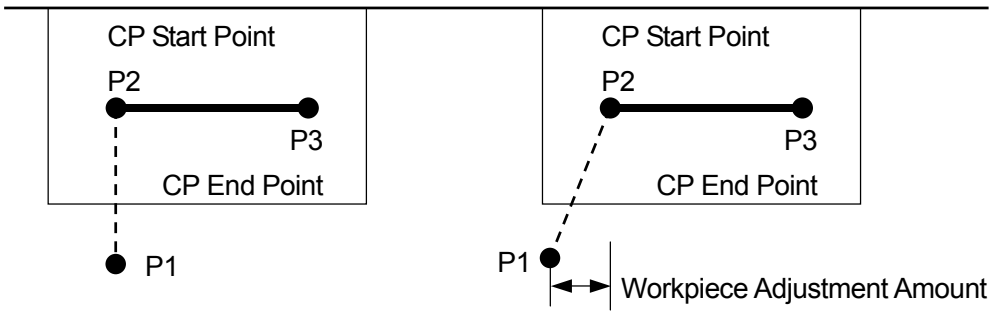
e.g.) Line dispense between P2-P3.

At P1, workpiece adjustment amount (workpiece offset value) is received from the sensor connected to COM.

The workpiece adjustment No. is made “6” and the tool is connected by the following setting.

“Dispensing start” : #handOut1=ON

“Dispensing end” : #handOut1=OFF



Point job data set in P1

declear str hosei inCom hosei,port1,10 #workAdj_X(6) = hosei	String type local variable “hosei” declaration. Receive workpiece adjustment from COM1 at “hosei”. The “hosei” value is assigned to #workAdj_X (6) (#workAdj_X(6)=X direction adjustment amount of workpiece adjustment 6)

Point job data set in P2 (Setting point of [Workpiece Adjustment])

set #handOut1	Dispensing start

Point job data set in P3

reset #handOut1	Dispensing end

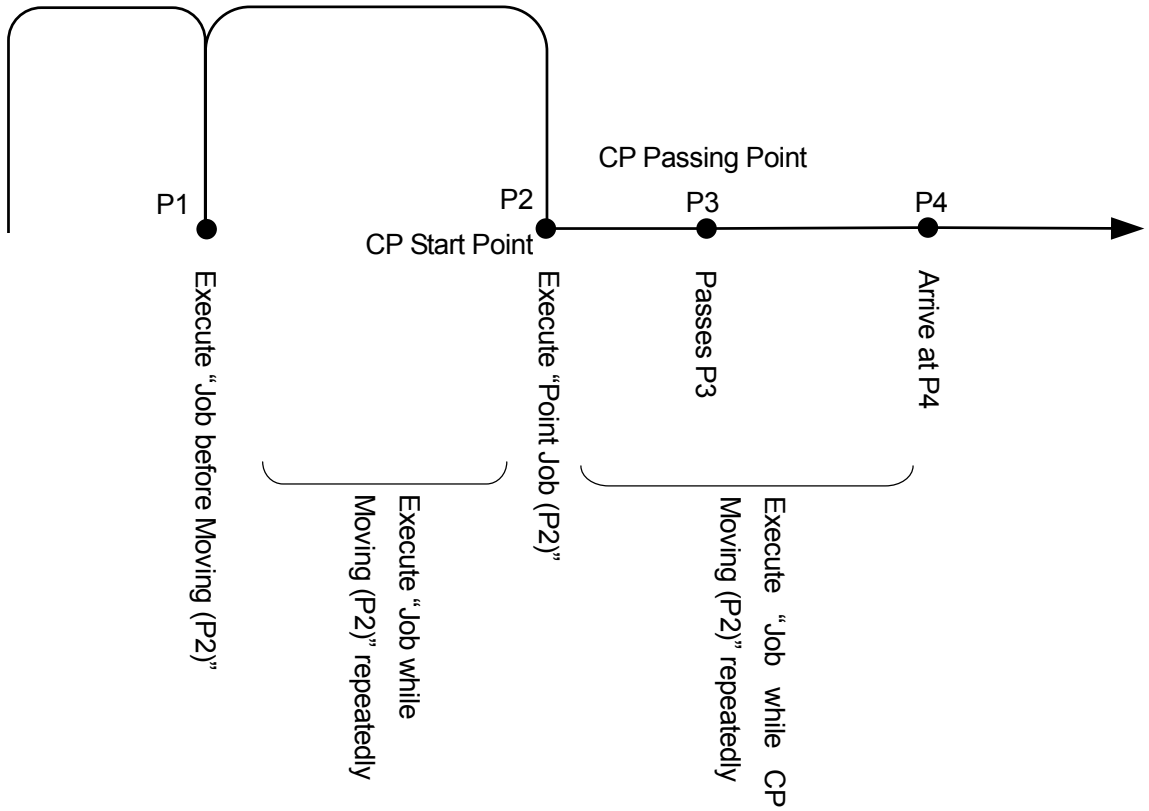
- When [Workpiece Adjustment] is set at the CP start point, it is valid until the tool unit reaches the CP end point.

Point coordinates: #point_X,#point_Y,#point_Z, #point_R,#point_TagCode

These variables hold the coordinates and tag code value of the running point. “Running point” is the point with point job data containing this variable set. When point job data containing this variable is set to a job before moving, job while moving, or job while CP moving, the current position of the tool center point and the value of this variable are different.

In the figure below, a job before moving set in P2 runs at point P1, but when the point job data set at job before moving includes this variable, the P2 coordinates are retained.

Also, this variable holds the original coordinates of the point. This value does not change even when [Workpiece Adjustment] and “#jogStartHight” are used.



Category	Identifier	Description
Current point coordinates	#point_X	Running point X coordinate value
	#point_Y	Running point Y coordinate value
	#point_Z	Running point Z coordinate value
	#point_R	Running point R coordinate value
	#point_TagCode	Running point tag code value

Given point coordinates:

#P_X, #P_Y, #P_Z, #P_R, #P_TagCode

These variables hold the coordinates and tag code value of a given point in the current program.
This variable holds the original coordinates of the point. This value does not change even when [Workpiece Adjustment] and "#jogStartHight" are used.

Category	Identifier	Description
Given point coordinate	#P_X (1 to last point count)	X coordinate value of given point in current program
	#P_Y (1 to last point count)	Y coordinate value of given point in current program
	#P_Z (1 to last point count)	Z coordinate value of given point in current program
	#P_R (1 to last point count)	R coordinate value of given point in current program
	#P_TagCode (1 to last point count)	Tag code value of given point in current program

Given point coordinates in given programs:

#prog_P_X, #prog_P_Y, #prog_P_Z,

#prog_P_R, #prog_P_TagCode

These variables hold the coordinates and tag code value of a given point in a given program.
This variable retains the original coordinates of the point. This value does not change even when [Workpiece Adjustment] and "#jogStartHight" are used.

Category	Identifier	Description
Coordinates of given point of given program	#prog_P_X (1 to 255, 1 to last point count)	X coordinate value of given point in given program
	#prog_P_Y (1 to 255, 1 to last point count)	Y coordinate value of given point in given program
	#prog_P_Z (1 to 255, 1 to last point count)	Z coordinate value of given point in given program
	#prog_P_R (1 to 255, 1 to last point count)	R coordinate value of given point in given program
	#prog_P_TagCode (1 to 255, 1 to last point count)	Tag code value of given point in given program

[Functions]

Robot functions

Built-in functions, that are built-in as robot functions, and user definition functions, which can be freely defined, can be used with this robot.
User definition functions are defined in the customizing mode. (Refer to operation manual “Features IV” for a description of the customizing mode.)
The following can be used as robot system built-in functions.

Category	Identifier	Description
num	currentMainProgNumber()	Currently running main program No.
num	currentSubProgNumber()	Currently running subprogram No.
num	currentPointNumber()	Currently running point No.
num	currentArmX()	Current arm coordinate X, “mm” units
num	currentArmY()	Current arm coordinate Y, “mm” units
num	currentArmZ()	Current arm coordinate Z, “mm” units
num	currentArmR()	Current arm coordinate R, “degree” units
num	numCOM(port#)	COM receive port data byte count

currentMainProgNumber()
Holds the main program number currently running.

currentSubProgNumber()
Holds the subprogram number currently running. When a subprogram is not run, this variable holds the main program number currently running.

currentPointNumber()
Holds the point number currently running. For work home, this variable is “0”.

currentArmX(),currentArmY(),currentArmZ()
Holds the current arm position (coordinate). (Absolute coordinate, “mm” units)

currentArmR()
Holds the number of rotations of the current R-axis (R-axis coordinate). (Absolute coordinate, “degree” units)

Arithmetic functions

The following can be used as arithmetic built-in functions.

x, y : Numeric value

n, m : Rounded integer value

Category	Identifier	Description
num	abs(x)	Absolute value
num	max(x,y)	Maximum value
num	min(x,y)	Minimum value
num	degrad(x)	Conversion from degree to radian (x*π/180)
num	raddeg(x)	Conversion from radian to degree (x*180/π)
num	sqrt(x)	Square root
num	sin(x)	Sine
num	cos(x)	Cosine
num	tan(x)	Tangent
num	atan(x)	Arc tangent
num	atan2(x,y)	Arc tangent
num	int(x)	Maximum integer which does not exceed x Ex: int (1.3)=1, int (-1.3)=-2
num	ip(x)	Integer part of x. sgn (x)*int (abs (x)) (When x is a negative number, sgn(x) becomes -1 and when x is a positive number, sgn(x) becomes +1.) Ex: ip (1.3)=1, ip (-1.3)=-1
num	fp(x)	Decimal part of x. x-ip (x) Ex: fp (1.3)=0.3, fp (-1.3)=-0.3
num	mod(x,y)	Value of x which makes y modulo. x-y*int (x/y)
num	remainder(x,y)	Remainder of dividing x by y. x-y*ip (x/y)
num	pow(x,y)	x to the power of y

String functions

The following can be used as string built-in functions.

x, y: Numeric value or Numeric variable

n, m: Numeric value made a certain digit or greater by rounding or truncation

a, b: String or string variable

Category	Type	Identifier	Description
String system	str	chr(x)	Returns a string (1 character) with the given character code.
	num	ord(a)	Returns the value of the top character code. Other characters are ignored.
	num	len(a)	Returns the string length (byte length). Does not handle multi byte codes.
	num	strPos(a,b)	Returns the first part string position that matches b in a.
	str	strMid(a,n,m)	Returns a character string from the n th to the m th characters, counting from the top of character string a.
	str	str(x)	Converts a numeric value to a decimal string.
	str	strBin(n,m)	Converts a numeric value to a binary string. m: Number of binary string digits
	str	strHex(n,m)	Converts a numeric value to a hexadecimal string. m: Number of hexadecimal string digits
	str	str1SI(x)	Rounds a numeric value to a 1 byte signed integer and converts it to a 1 byte string. (1 byte Signed Integer)
	str	str2SIBE(x)	Rounds a numeric value to a 2 byte signed integer and converts it to a 2 byte string big endian. (2 byte Signed Integer Big Endian)
	str	str2SILE(x)	Rounds a numeric value to a 2 byte signed integer and converts it to a 2 byte string little endian. (2 byte Signed Integer Little Endian)
	str	str4SIBE(x)	Rounds a numeric value to a 4 byte signed integer and converts it to a 4 byte string big endian. (4 byte Signed Integer Big Endian)
	str	str4SILE(x)	Rounds a numeric value to a 4 byte signed integer and converts it to a 4 byte string little endian. (4 byte Signed Integer Little Endian)
	str	str4FBE(x)	Regards a numeric value as a floating decimal and converts it to a 4 byte string big endian. (4 byte Float Big Endian)
	str	str4FLE(x)	Regards a numeric value as a floating decimal and converts it to a 4 byte string little endian. (4 byte Float Little Endian)
	str	str8DBE(x)	Regards a numeric value as a double floating decimal and converts it to an 8 byte string big endian. (8 byte Double Big Endian)
	str	str8DLE(x)	Regards a numeric value as a double floating decimal and converts it to 8 byte string little endian. (8 byte Double Little Endian)
	num	val(a)	Regards a string as a decimal string and converts it to a numeric value.

x, y: Numeric value or Numeric variable

n, m: Numeric value made a certain digit or greater by rounding or truncation

a, b: String or string variable

Category	Type	Identifier	Description
String system	num	valBin(a)	Regards a string as a binary string (list of “0” and “1”) and converts it to a numeric value.
	num	valHex(a)	Regards a string as a hexadecimal string (list of “0” to “1”, “A” to “F”, or “a” to “f”) and converts it to a numeric value.
	num	val1SI(a)	Regards the top character as a 1 byte signed integer and converts it. (1 byte Signed Integer)
	num	val2SIBE(a)	Regards the top 2 characters as a 2 byte signed integer big endian and converts it. (2 byte Signed Integer Big Endian)
	num	val2SILE(a)	Regards the top 2 characters as a 2 byte signed integer little endian and converts it. (2 byte Signed Integer Little Endian)
	num	val4SIBE(a)	Regards the top 4 characters as a 4 byte signed integer big endian and converts it. (4 byte Signed Integer Big Endian)
	num	val4SILE(a)	Regards the top 4 characters as a 4 byte signed integer little endian and converts it. (4 byte Signed Integer Little Endian)
	num	val4FBE(a)	Regards the top 4 characters as a floating decimal big endian and converts it. (4 byte Float Big Endian)
	num	val4FLE(a)	Regards the top 4 characters as a floating decimal little endian and converts it. (4 byte Float Little Endian)
	num	val8DBE(a)	Regards the top 8 characters as a double floating decimal big endian and converts it. (8 byte Double Big Endian)
	num	val8DLE(a)	Regards the top 8 characters as a double floating decimal little endian and converts it. (8 byte Double Little Endian)
	num	valSum(a)	Returns the sum of the string code added from top to bottom.
	num	valCRC(a)	Remainder of division of string assumed to be a bit string divided by generation polynomial $X^{16}+X^{12}+X^5+1$
	str	bitNot(a)	Bit NOT
	str	bitAnd(a,b)	Bit logical AND
	str	bitOr(a,b)	Bit logical OR
	str	bitXor(a,b)	Bit exclusive logical OR

[ON/OFF Output Control]

Outputting to I/O: set, reset, pulse, invPulse

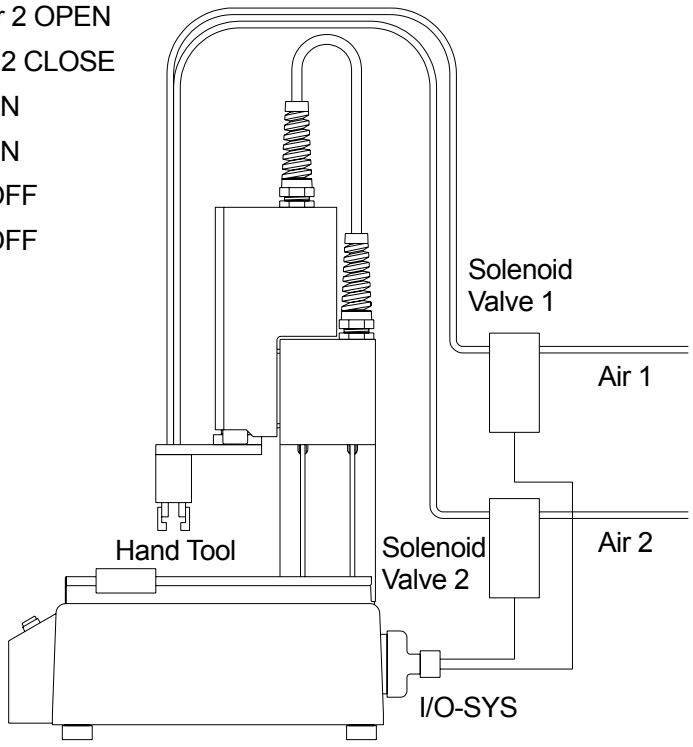
This section explains commands to be output to a tool (output to the I/O.) These commands belong to the category [ON/OFF Output Control.]

Command Category	Command	Parameter		Job
ON/OFF Output Control	set	Output Destination		ON output to a specified output destination
	reset	Output Destination		OFF output to a specified output destination
	pulse	Output Destination	Pulse Width	ON pulse output with a specified width to a specified output destination
	invPulse	Output Destination	Pulse Width	OFF pulse (inverting pulse) output with a specified width to a specified output destination

Example)

In this example, a hand tool is connected to the robot as shown in the following figure.

- Hand Tool OPEN ← Air 1 CLOSE & Air 2 OPEN
- Hand Tool CLOSE ← Air 1 OPEN & Air 2 CLOSE
- Air 1 OPEN ← Solenoid Valve 1 ON
- Air 2 OPEN ← Solenoid Valve 2 ON
- Air 1 CLOSE ← Solenoid Valve 1 OFF
- Air 2 CLOSE ← Solenoid Valve 2 OFF
- Solenoid Valve 1 ON ← #sysOut15 ON
- Solenoid Valve 2 ON ← #sysOut16 ON
- Solenoid Valve 1 OFF ← #sysOut15 OFF
- Solenoid Valve 2 OFF ← #sysOut16 OFF



Therefore,

Hand Tool OPEN ← #sysOut15 OFF & #sysOut16 ON

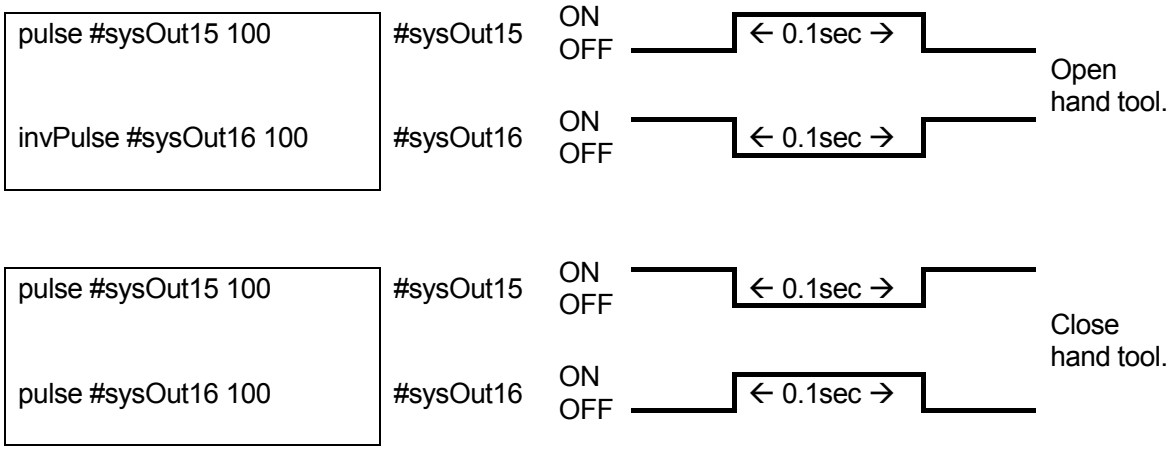
Hand Tool CLOSE ← #sysOut15 ON & #sysOut16 OFF

Below are the output commands to open/close the hand tool.

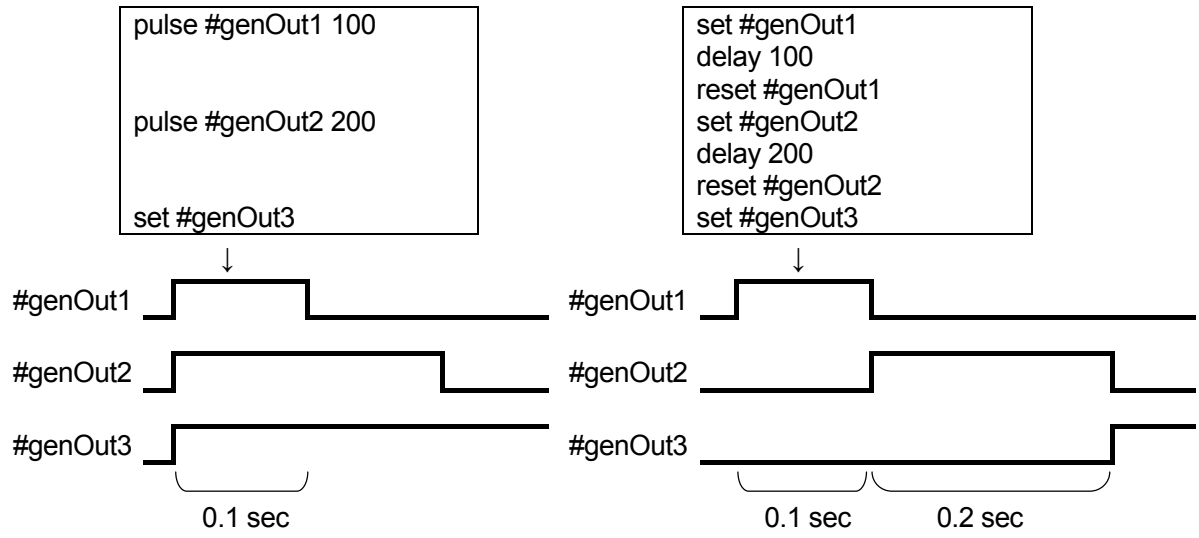
set #sysOut15 reset #sysOut16	#sysOut16 ON output #sysOut15 OFF output	Open hand tool.
set #sysOut15 reset #sysOut16	#sysOut16 ON output #sysOut15 OFF output	Close hand tool.

- The command [set] continues to output an ON signal unless the command [reset] is received.

Below are the output commands to open/close the hand tool using the pulse.



- The commands [pulse] and [invPulse] move on to the next command before the pulse stops.
  In the following example 2 point job data have different results:



  delay 100 = Stand by for 0.1 second in place.

- You can set the pulse width for the commands [pulse] and [invPulse] using variables or expressions.

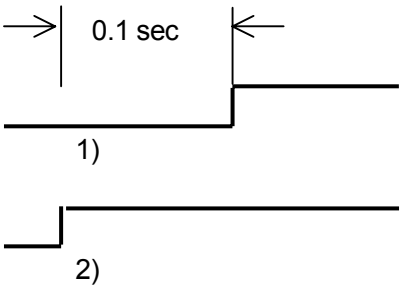
Outputting after X second: delaySet, delayReset

The commands “delaySet” and “delayReset” are used to output ON/OFF signals to a specified output destination after a specified period of time.
The delay time can be set from 0.001 sec to 9999.999 sec.

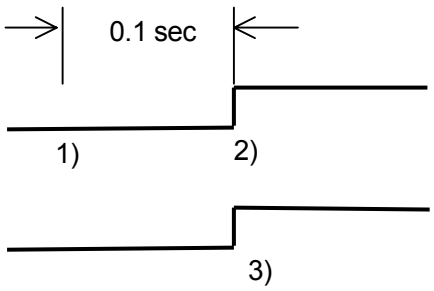
Command Category	Command	Parameter		Job
ON/OFF Output Control	delaySet	Output Destination	Delay Time	ON output after specified delay time
	delayReset	Output Destination	Delay Time	OFF output after specified delay time

The commands “delaySet” and “delayReset” move on to the next command before the pulse stops. Timing of the next command execution differs from the case where signals are output by set/reset after “waitCondTime.”

Example) delaySet
1) delaySet #sysOut2 100
2) set #sysOut1
3) ・・・・・・



Example) waitCondTime/set
1) waitCondTime 100
2) set #sysOut1
3) set #sysOut2
4) ・・・・・・



- Delay Time can be set using variables or expressions.

Sounding an alarm buzzer: onoffBZ

A point job sounds an alarm buzzer.

Command Category	Command	Parameter	Job
ON/OFF Output Control	set	Output Destination (BZ)	Sounds an alarm buzzer.
	reset	Output Destination (BZ)	Stops an alarm buzzer.
	onoffBZ	ON Time, OFF Time	Sounds an alarm buzzer off and on.

If these commands “set/onoffBZ” are executed, an alarm buzzer continues to sound until the “reset”command is executed.

- ON Time and OFF Time for the command “onoffBZ” can be set using variables or expressions.

Blinking the LED (Green): onoffGLED

- The following commands are valid for the CARTESIAN series only.

The following explains how to turn ON or blink the LED light on the front body using point job commands.

Command Category	Command	Parameter	Job
ON/OFF Output Control	set	Output Destination (GLED)	Turns the LED (Green) ON.
	reset	Output Destination (GLED)	Turns the LED (Green) OFF.
	onoffGLED	ON Time, OFF Time	Blinks the LED (Green.)

After the onoffGLED command, which turns ON or blinks the LED (Green), is executed, the LED (Green) stays ON or keeps blinking until the reset command, which turns the LED (Green) OFF, is executed.

- ON Time and OFF Time for “onoffGLED” command can be set using variables or expressions.

Blinking the LED (Red): onoffRLED

- The following commands are valid for the CARTESIAN series only.

The following explains how to turn ON or blink the LED light on the front body using point job commands.

Command Category	Command	Parameter	Job
ON/OFF Output Control	set	Output Destination (RLED)	Turns the LED (Red) ON.
	reset	Output Destination (RLED)	Turns the LED (Red) OFF.
	onoffRLED	ON Time, OFF Time	Blinks the LED (Red.)

After the onoffRLED command, which turns ON or blinks the LED (Red), is executed, the LED (Red) stays ON or keeps blinking until the reset command, which turns the LED (Red) OFF, is executed.

- ON Time and OFF Time for “onoffRLED” command can be set using variables or expressions.

Outputting values from I/O: dataOut,dataOutBCD

The optional numeric values “0 to 999,999,999” or tag codes can be output to the I/O or the Boolean free variables (#mv(1~99), #mkv(1~99).)

Command Category	Command	Parameter			Job
ON/OFF Output Control	dataOut	Output Value	Output Destination	Output Width	Outputs values from the I/O.
	dataOutBCD	Output Value	Output Destination	Output Width	Outputs values in BCD from the I/O.

- Using tag code output, you can output different values using the same point job data if you set different values as tag codes to multiple points.
- Output Values and Output Width can be set using variables or expressions.

You need to set the following 2 parameters in addition to Output Value for the commands “dataOut” and “dataOutBCD”.

- Output Width: The number of I/Os to be used for output
- Output Destination: The smallest number between I/Os to be used for output

e.g) If you use #genOut8 to 10, the Output Destination is “8.”

- The serial I/Os are used for the commands “dataOut” and “dataOutBCD”. You cannot use I/Os that are not serial I/Os.

Example:

<Setting>	<Command>	<Output> 6=110 (binary)
Output Value: 6		#genOut8: 0 (OFF)
Output Width: 3	dataOut 6,3,#genOut8	#genOut9: 1 (ON)
Output Destination: #genOut8		#genOut10: 1 (ON)

- If an Output Value does not match a specific Output Width, the upper digit will be truncated.

Example:

<Setting>	<Command>	<Output> 14=1110 (binary)
Output Value: 14		#genOut8: 0 (OFF)
Output Width: 3	dataOut 14,3,#genOut8	#genOut9: 1 (ON)
Output Destination: #genOut8		#genOut10: 1 (ON)
		: 1 (truncation)

- Output Width can be set up to “31.” However, the two I/O’s cannot be combined.

[If Branch, Wait Condition]

if Branch: if, then, else, endIf

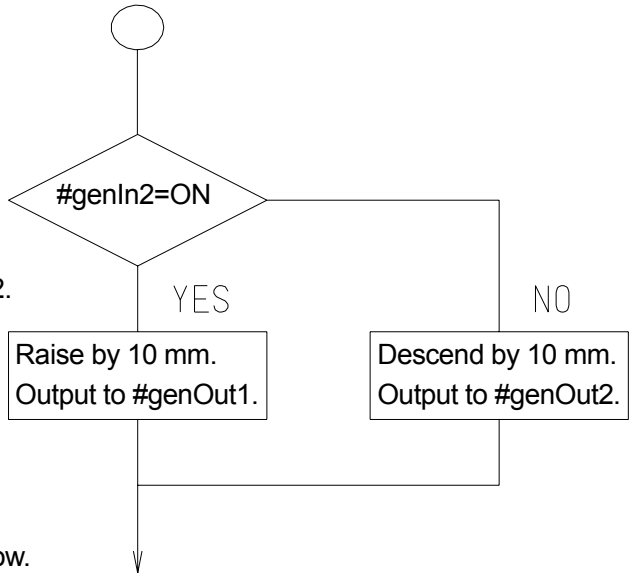
This section explains point job data commands for executing different jobs according to certain conditions. These commands belong to the category [if Branch, Wait Condition.]

Command Category	Command	Parameter	Job
if Branch, Wait Condition	if	-	if Branch
	then	-	Executes the following commands if true.
	else	-	Executes the following commands if false.
	endIf	-	End of if Branch

- Be sure to put the commands for the Condition after "if."

Examples of if, then, else and endIf

Example 1: If #genIn2 is ON, raise the Z axis by 10 mm and output a pulse to #genOut1.
If #genIn2 is not ON, descend the Z axis 10 mm and output a pulse to #genOut2.

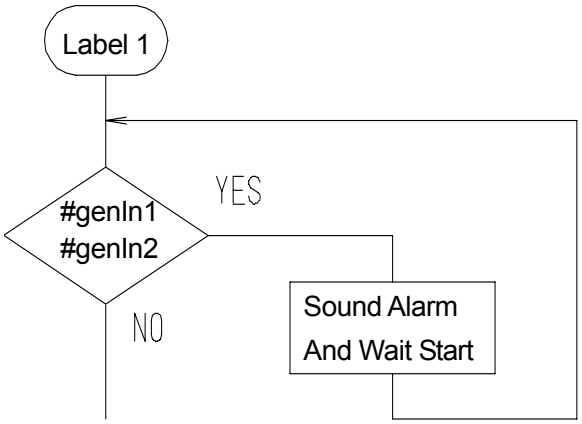


The Commands for Example 1 are shown below.

```
if
  ld #genIn2
then
  upZ 10,20
  pulse #genOut1,200
else
  downZ 10,20
  pulse #genOut2,200
endIf
```

If the following condition is true, advance to then. If false, advance to else.
#genIn2 = ON (Condition)
If the Condition is true, execute the following commands.
Raise the Z axis by 10 mm at the speed of 20 mm/sec,
And output ON pulse to #genOut1. (The pulse width is 0.2 seconds.)
If the Condition is false, execute the following commands.
Descend the Z axis by 10 mm at the speed of 20 mm/sec,
and output ON pulse to #genOut2 (The pulse width is 0.2 seconds.
End of if Branch

Example 2: If #genIn1 and #genIn2 are both ON, sound an alarm buzzer and stand by until a start instruction is received.
If either #genIn1 or #genIn2 are not ON, advance to the next job.



The commands for Example 2 are shown below.

Label 1	(A destination mark for jump command)
if	If the following condition is true, advance to then. If false, advance to the next of endIf.
ld #genIn1	#genIn1=ON (Condition 1)
and #genIn2	And #genIn2=ON (Condition 2)
then	If the Conditions are true, execute the following commands.
waitStartBZ	Sound an alarm buzzer and stand by in place until a start instruction is received.
jump L1	Jump to [Label 1] when a start instruction is received.
endIf	End of If Branch

- It is not necessary that both "then …" and "else …" exist at the same time. However, an IF command without a corresponding "endIf" command is recognized as an error.
- The command lines for "waitCondTime" "timeUp" … "endWait", "if" … "endIf" are indented. (See below.)

```
waitCondTime 200
  ld #genIn2
timeUp
  set genOut2
    if
     ld #genIn1
    then
        downZ 20,20
        waitCondTime 200
            ld #genIn4
        timeUp
            waitStartBZ
        endWait
    endIf
endWait
```

3rd level
2nd level
1st level

Be sure not to exceed the 9th level of the indent.

If point job data including a line with the indent exceeding 9th level, it will recognize an error in running and the message "Error on point job" will be displayed.

If "timeUp" or "endWait" precedes "waitCondTime" or if "then", "else" or "endIf" proceeds "if", it will be also recognized as an error and the message "Error on point job" will be displayed.

Wait Condition: waitCond,waitCondTime,timeUp,endWait

This section explains the point job data commands for waiting until the sensor (connected to #genIn2) comes ON. These commands belong to the category [Wait Condition.]

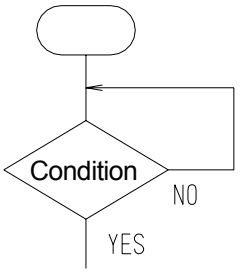
Command Category	Command	Parameter	Job
Wait Condition	waitCondTime	Period for Time Out	Waits for conditions for a certain period.
	timeUp	-	Executes when time is up.
	endWait	-	End of WAIT command
	waitCond	-	Waits for conditions.

- Wait Condition commands are invalid at the CP Passing Point or a point whose Base Type is CP Passing Point.
- Be sure to put condition commands after “waitCond” or “waitCondTime.”

waitCond … endWait : Wait until conditions are met.

e.g.: A workpiece exists. → Sensor (#genIn2) ON

A workpiece does not exist. → Sensor (#genIn2) OFF



waitCond ld #genIn2 endWait	Stand by in place until the following conditions are met. #genIn2=ON (Condition) End of the condition line.

waitCondtime … timeUp … endWait: Wait for the specified period of time until conditions are met.

e.g.: If workpieces do not come within 30 seconds, it is recognized as an error, an external lamp (connected to #genOut2) comes ON, and the robot stands by until a start instruction comes.

To re-start running the robot, resolve the problem and press the start button.

waitCondTime 3000 ld #genIn2 timeUp set #genOut2 waitStartBZ reset #genOut2 endWait	Wait for 3 seconds until the following conditions are met. #genIn2=ON (Condition) If the conditions are not met within 3 seconds, Output ON signal to #genOut2, Stand by in place until a start instruction comes. Output OFF signal to #genOut2 when a start instructionis received. End of the line for commands if conditions are not met after 3 seconds.

- [endWait] and [timeUp] cannot be used alone.
- A period for Time Out of “waitCondTime” can be set using variable and expressions.

Example)

```
declear num wtime
if
  ld #genIn3
then
  wtime = 3000
else
  wtime = 1000
endIf
waitCondTime wtime
  ld #genIn2
timeUp
  set #genOut2
  waitStartBZ
  reset #genOut2
endWait
```

Declare the local variable “wtime.”
If
  #genIn3=ON
then
  Assign 3000 to”wtime.”
if
  Assign 1000 to “wtime.”

Wait for 3 seconds/1 second until the following conditions are met.
  #genIn2=ON (Condition)
If the conditions are not met within 3 seconds/1 second,
  Output ON signal to #genOut2,
  Stand by in place until a start instruction comes.
  When a start instruction comes, output OFF signal to #genOut2.
End of the command line if the conditions are not met within 3 seconds/1 second.

[Condition]

Condition Settings: ld, ldi, and, ani, or, ori, anb, orb

The following describes the condition commands placed after the If Branch, Wait Condition (if, waitCond, waitCondTime) commans. The command category it belongs to is [Condition.]

Command Category	Command	Parameter	Job
Condition	ld	Boolean variable or expression	ON input
	ldi	Boolean variable or expression	OFF input
	and	Boolean variable or expression	Serial ON input
	ani	Boolean variable or expression	Serial Off input
	or	Boolean variable or expression	Parallel ON input
	ori	Boolean variable or expression	Parallel OFF input
	anb		Block serial connection
	orb		Block parallel connection

I/O-SYS output (#sysOut), I/O-1 output (#genOut), I/O-H output (#handOut), system flag (#sysFlag), internal relay (#mv), keep relay (#mkv), and pallet flag, as well as I/O-SYS input (#sysIn), I/O-1 input (#genin), and I/O-H input (#handIn), can be given as command parameters.
Comparison operation expressions can also be used. Variables and functions other than the above parameters can also be used in comparison operation expressions.

Comparison operation expression	Meaning
○ = □	Equal □ to ○
○ < □	□ greater than ○
○ > □	□ less than ○

Comparison operation expression	Meaning
○ <= □ ○ =< □	□ greater than or equal to ○
○ >= □ ○ => □	□ less than or equal to ○
○ <> □ ○ >< □	Not equal

A comparison operation expression must always start from an ld or ldi command line. If it is only an independent ON (true) or OFF (false) condition, it is a 1 line command, but when multiple conditions are connected with and, or, etc., it becomes a multi-line command string.
Expressions can also be used in conditional operations. In this case, the result of the expression is judged as 0 (flag) or nonzero (true).

ld: ON input

```
waitCond
  ld #genIn2
endWait
```

Waits in place until the following condition is met.
#genIn2=ON (condition)
End of condition line

ldi: OFF input

```
waitCond
  ldi #genIn2
endWait
```

Waits in place until the following condition is met.
#genIn2=OFF (condition)
End of condition line

and: Series ON input

```
waitCond
  ld #genIn1
  and count>=10
endWait
```

Waits in place until the following conditions are met.
#genIn1 is ON (condition 1)
and count value is 10 or greater (condition 2)
End of condition line

- "count" is a variable.

ani: Series OFF input

```
waitCond
  ldi #genIn1
  ani count<=10
endWait
```

Waits in place until the following conditions are met.
#genIn1 is OFF (condition 1)
and count value is 10 or less (condition 2)
End of condition line

or: Parallel ON input

```
waitCond
  ld #genIn1
  or #genIn2
endWait
```

Waits in place until the following condition is met.
#genIn1 is ON (condition 1)
or #genIn2 is ON (condition 2).
End of condition line

ori: Parallel OFF input

```
waitCond
  ldi #genIn1
  ori #genIn2
endWait
```

Waits in place until the following condition is met.
#genIn1 is OFF (condition 1)
or #genIn2 is OFF (condition 2)
End of condition line

anb: Block series connection

```
waitCond
  ld count>=10
  or flag
  ldi #genIn1
  ani #genIn2
  anb
endWait
```

Waits in place until the following conditions are met.

- count is 10 or greater or flag is ON — Condition 1
- #genIn1 is OFF and #genIn2 is OFF — Condition 2

Condition 1 is true and condition 2 is also true

End of condition line

orb: Block parallel connection

```
waitCond
  ld count>=10
  or flag
  ldi #genIn1
  ani #genIn2
  orb
endWait
```

Waits in place until the following conditions are met.

- Count is 10 or greater or flag is ON — Condition 1
- #genIn2 is OFF and #genIn2 is also OFF — Condition 2

Condition 1 or condition 2 is true.

End of condition line

- When there is “anb” and “orb” but no corresponding “ld” or “ldi”, “Error on point job” is displayed and an error is recognized.

[Delay, Data In, Wait Start]

Time Delay: delay

This section explains the point job data command for controlling time delay.

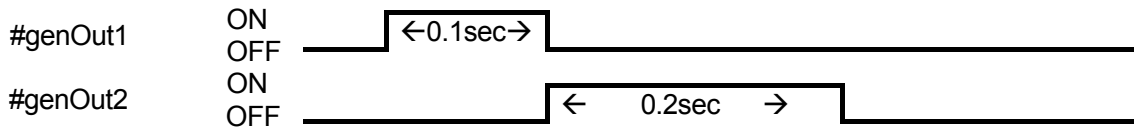
Command Category	Command	Parameter	Job
Delay, Data In, Wait Start	delay	Delay Time	Stand by in place for a specified period of delay time.

- The command “delay” is invalid at a CP Passing Point and at a point whose Base Type is CP Passing Point.

delay : Delay for the specified period of time.

Example)

set #genOut1	Output ON signal to #genOut1,
delay 100	Delay for 0.1 sec.
reset #genOut1	Output OFF signal to #genOut1.
set #genOut2	Output ON signal to #genOut2.
delay 200	Delay for 0.2 sec.
reset #genOut2	Output OFF signal to #genOut2.



Delay Time can be set using variable or expression instead of values.

Example)

declear num wtime	Declare the local variable “wtime.”
if	If
ld #genIn1	#genIn1=ON
then	then
wtime = 100	Assign 100 to “wtime.”
else	If not
wtime = 200	Assign 200 to “wtime.”
endIf	
set #genOut1	Output ON signal to #genOut1.
delay wtime	Delay for 0.1 sec/0.2 sec.
reset #genOut1	Output OFF signal to #genOut1.

Waiting for a start instruction: waitStart, waitStartBZ

This section explains the point job data commands to stop running until a start instruction is received.

Command Category	Command	Parameter	Job
Delay, Data In, Wait Start	waitStart	-	Stands by in place until a start instruction is received.
	waitStartBZ	-	Stands by in place while sounding an alarm buzzer until a start instruction is received.

- The commands “waitStart” and “waitStartBZ” are invalid at the CP Passing Point or a point whose Base Type is CP Passing Point.

waitStart: Wait for start.

Example)

```
set #genOut1
waitStart
reset #genOut1
```

Outputs an ON signal to #genOut1.
Stands by in place until a start instruction comes.
Outputs an OFF signal to #genOut1 (when a start instruction is received.)

waitStartBZ: Wait for start (alarm buzzer)

For Example: If #genIn1 does not come ON within 2 seconds, it is recognized as an error, genOut2 (connected to an external alarm/error lamp) comes ON and the robot “stands by for start while sounding an alarm buzzer.”

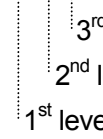
When an operator resolves the problem and sends a start instruction, the OFF signal will be output to #genOut2 and the operation will start from Point 05.

```
waitCondTime 2000
  ld #genIn1
timeUp
  upZ 50,20
  set #genOut2
  waitStartBZ
  reset #genOut2
  goPoint PTP3,5
endWait
```

Wait for #genIn1 to go ON for 2 seconds.
If #genIn1 does not come ON within 20 seconds,
Raise the Z axis 50 mm (at the speed of 20 mm/sec),
Output ON signal to #genOut2,
Sound an alarm buzzer and stand by in place until a start instruction is received.
Output OFF signal to #genOut2 (when a start instruction comes),
Go to Point 05.
End of the comannd which will be executed if #genIn1 does not come ON within 20 seconds.

- If using “waitCondTime” “timeUp” … “endWait” or “if” … ”endIf”, command lines are indented. (See below.)

```
waitCondTime 200
  ld #genIn2
timeUp
    set genOut2
    if
      ld #genIn1
    then
      downZ 20,20
      waitCondTime 200
          ld #genIn4
      timeUp
          waitStartBZ
      endWait
    endIf
endWait
```



Be sure not to exceed the 9th level of the indent.

If point job data including a line with an indent exceeding the 9th level, it will recognize a running error and the message “Error on point job” will be displayed.

If “timeUp” or “endWait” precedes “waitCondTime” or if “then”, “elese” or “endIf” precedes “if”, it will be also recognized as an error and the message “Error on point job” will be displayed.

Inputting from I/O: dataIn, dataInBCD

Read out a value from the I/O or Boolean variable (#mv (1 to 99), #mkv (1 to 99)) and assign it to the specified variable.

Command Category	Command	Parameter			Job
Delay, Data In, Wait Start	dataIn	Variable to assign to	Input Destination	Input Width	Read out numeric data from I/O.
	dataInBCD	Variable to assign to	Input Destination	Input Width	Read numeric data in BCD from I/O.

- BCD = Binary-Coded Decimal
- Read out width can be set using variables or expressions.

For the commands “dataIn” and “dataInBCD”, you need to set the following 2 parameters in addition to a variable to which an input value is assigned.

- Input Width: The number of I/Os to be used to input
- Input Destination: The smallest number between I/Os to be used to input

e.g.) If you use #genIn3 to 10, the Read Out Source is “3.”

- The serial I/Os from Input Destination to Input Width are used. You cannot use I/Os that are not serial I/O’s.

Example)

```
declear numeric code
dataIn code,#genIn3,8
```

Declare the local variable “code.”
Read out data from #genIn3 (I/O-1) to #genIn10 as a value and assign it to “code.”

```
declear numeric code
dataInBCD code,#genIn3,8
```

Declare the local variable of “code.”
Read out data from #genIn3 (I/O-1) to #genIn10 as BCD value and assign it to “code.”

Status of I/O-1

#genIn3	#genIn4	#genIn5	#genIn6	#genIn7	#genIn8	#genIn9	#genIn10
OFF	OFF	OFF	ON	OFF	OFF	ON	OFF

Input Width 8 →

In the above case, the values of “code” are as below.

For the command “dataIn”, the value of “code” is 18.

For the command “dataInBCD”, the value of “code” is 12.

Input Width can be set up to “31.” However, it cannot be extended to different I/O.

- Input Width can be set up to “31.” However, the two I/Os cannot be combined.

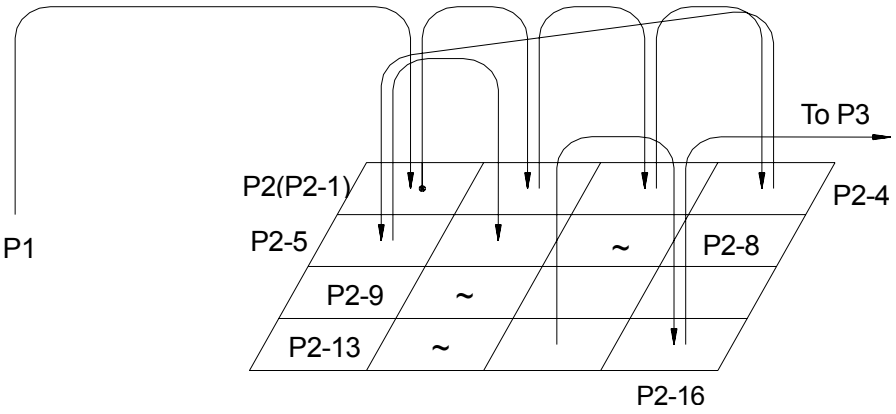
[Pallet Control]

Pallet Command: loopPallet, resPallet, incPallet

There are two methods for updating the pallet counter. One is [Auto Increment], which increases the counter automatically (the arm will proceed to the next position on the pallet), and the other is [Increment by Point Job], which will not increase the counter (that is, the tool unit will not move to the next position on the pallet) unless the point job specifies it.

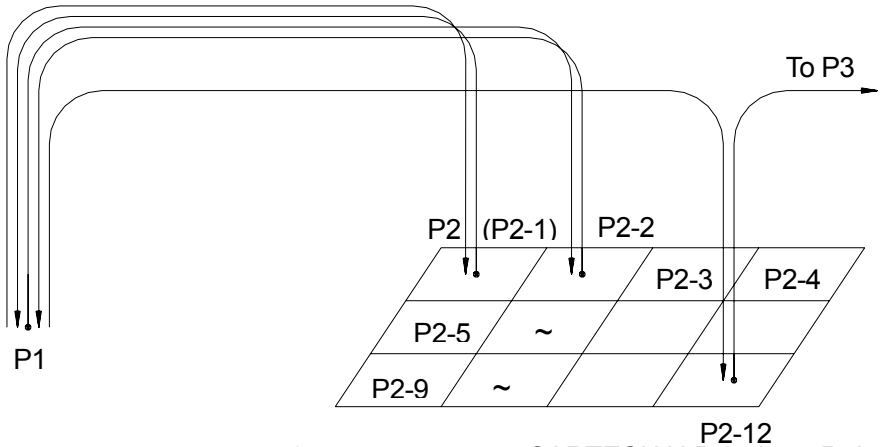
[Auto Increment] does not need a point job command to control the pallet operation. The tool unit will automatically move to the next point and update the pallet counter. However, on the [Auto Increment] pallette, the tool unit can only move in order of P2-1, P2-2, P2-3 and so on as shown below.

Example) Pallet of Auto Increment



[Plane Pallet]
Auto Increment
4 rows, 4 columns

Example) Pallet of Increment by Point Job



[Plane Pallet]
Increment by Point Job
4 rows, 3 columns

During the pallet operation [Increment by Point Job], the tool unit can move as shown in the figure on the previous page.
The tool unit returns to P1 before it moves to the next position each time. (P1 → P2 (P2-1) → P1 → P2-2 → P1 → P2-3, and so on.)

Below are the pallet commands used for [Increment by Point Job.]

Command Category	Command	Parameter	Job
Pallet	loopPallet	Pallet Number, go Point Number	Increase the counter one by one. Unless the counter has reached the maximum number, the arm will move to the specified point.
	resPallet	Pallet Number	Reset the counter to "0."
	incPallet	Pallet Number	Increase the counter by one.

The following 2 variables are unique to pallet control.

[paletteFlag(n)]: Boolean variable which has the following contents.
The counter of Pallet (Number n) is at the maximum=ON (true)
The counter of Pallet (Number n) is not at the maximum=OFF (false)
[palletCount(n)]: Numeric type variable which has the counter value of Pallet (Number n)

In the following example of point job data, the robot picks the workpiece up from P1 (set #genout1) and places it at P2 (reset #genOut1) on the pallet of [Increment by Point Job] as shown in the figure.

Point Job Data (for P1 setting)

set #genOut1

Holds (picks up) the workpiece.

Point Job Data (for P2 setting)

reset #genOut1 loopPallet 10,1

Releases (places) the workpiece.
Increases the counter of Pallet 10 by one.
If the counter reaches the maximum, it goes on to the next command. In this case, the point job ends because there is no next command.
If the counter is not at maximum, it moves to P1.

- Shifting by "loopPallet" (shift to P1 in case of point job data for the above P2 setting) is complied with "PTP Condition" in the program data.

Below are the pallet commands for using the command “incPallet (Increase the specified pallet counter by one)” instead of “loopPallet.”

<Use “incPallet” instead of “loopPallet”>

reset #genOut1
incPallet 10
if
  ld #palletFlag(10)
else
  goPoint PTP3,1
endif
	Hold (pick up) the workpiece.
	Increase the counter of Pallet 10 by one.
	if
	  The counter of Pallet 10 does not reach the maximum,
	
	  Go to P1 (comply with PTP Condition03.)

- If you use the command “loopPallet”, the arm shifts to a specified point complying with PTP Condition (Program data.) If you use the command “incPallet”, you can use the command “goPoint” or “goRPoint” and select PTP Condition (Additional Function Data.)

If you use the command “incPallet”, a pulse can also be output everytime the arm shifts to P1.

reset #genOut1
incPallet 10
if
  ld #paletteFlag(10)
else
  pulse #genOut5,200
  goPoint PTP0,1
endif
	Hold (pick up) the workpiece.
	Increase the counter of Pallet 10 by one.
	If
	  The counter of Pallet 10 does not reachthe maximum,
	  Output a pulse,
	  Shift to P1.

Pallet Number (also go Point Number in case of “loopPallet”) can be set using expressions.

Example)

declear num pal
if
  ld #genIn3
then
  pal = 5
else
  pal = 6
endIf
reset #genOut1
loopPallet pal,1
	Declare the local variable “pal.”
	If
	  #genIn3=ON
	then
	  Assign 5 to “pal.”
	If not
	  Assign 6 to “pal.”
	
	Hold (pick up) the workpiece.
	Increase the counter of Pallet 5/6 by one,
	go on to the next command if the counter reaches the maximum. (In this case, the point job ends because there is no next command.)
	Shift to P1 if the counter does not reach the maximum.

[Execution Flow Control]

Subroutine call of type setting job: callBase

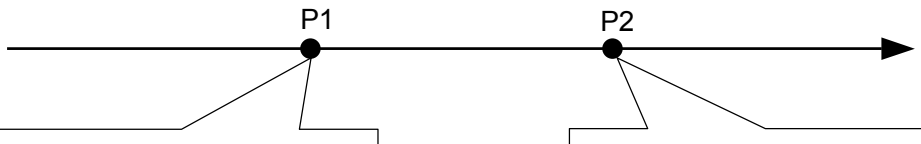
When a point job, etc. is set at a user definition type point created in the customizing mode, the point job, etc. added to the type is not executed.
Also, when an additional function is set at user definition type, and a function of the same type but different number is set at the point, the function data number of the function set at the point has priority.

e.g.) Assume there is the user definition type showed to the right. At this time, the point job data which is executed at operation at P1 and P2 in the below figure becomes:

Title	: Pick-up point
Base type	: PTP drive point
Job before moving	: Yes
Job while moving	: Yes
Point job	: Yes
Additional function	: PTP condition 1

P1) Job before moving : Point job data 5
Job while moving : Point job data 6
Point job : Point job data 7
Additional function : PTP condition 9
Execute condition 1

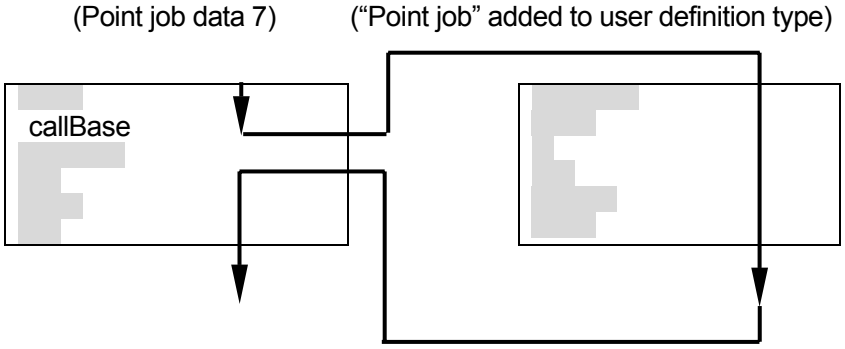
P2) Job before moving : "Job before moving" added to user definition type
Job while moving : "Job while moving" added to user definition type
Point job : "Point job" added to user definition type
Additional function : PTP condition 1
Execute condition 1



Point 01
Type: Pick-up point
Job before moving No.: 5
Job while moving No.: 6
Point job No.: 7
Additional function: PTP condition 9

Point 02
Type: Pick-up point
Job before moving No. : 0 (none)
Job while moving No. : 0 (none)
Point job No. : 0 (none)
Additional function : None

In these cases, when the “callBase” command is used in the point job data set at a user definition type point, a subroutine of the point job, etc. added to the type can be called. When the “callBase” command is used in point job data 7 of the example, a subroutine of the command string for the “point job” added to the user definition type of point job data 7 is called when a point job of P1 is executed.



Command category	Command	Parameters	Job
Execution flow control	callBase		Calls and executes the job command string added to the type at the user definition type point.

- The “callBase” command is invalid at the CP Passing Point or a point whose Base Type is CP Passing Point.

The “callBase” command calls a subroutine of the job command string added to the type. Therefore, when the “callBase” command is executed in a job before moving, job while moving, and job while CP moving, a subroutine command string for the job before moving, job while moving, and job while CP moving added to each type is called.
In the case of the example, when the “callBase” command is used at point job data 5, a subroutine of the command string for the job before moving added to the type at point job data 5 is called when P1 job before moving job is executed.

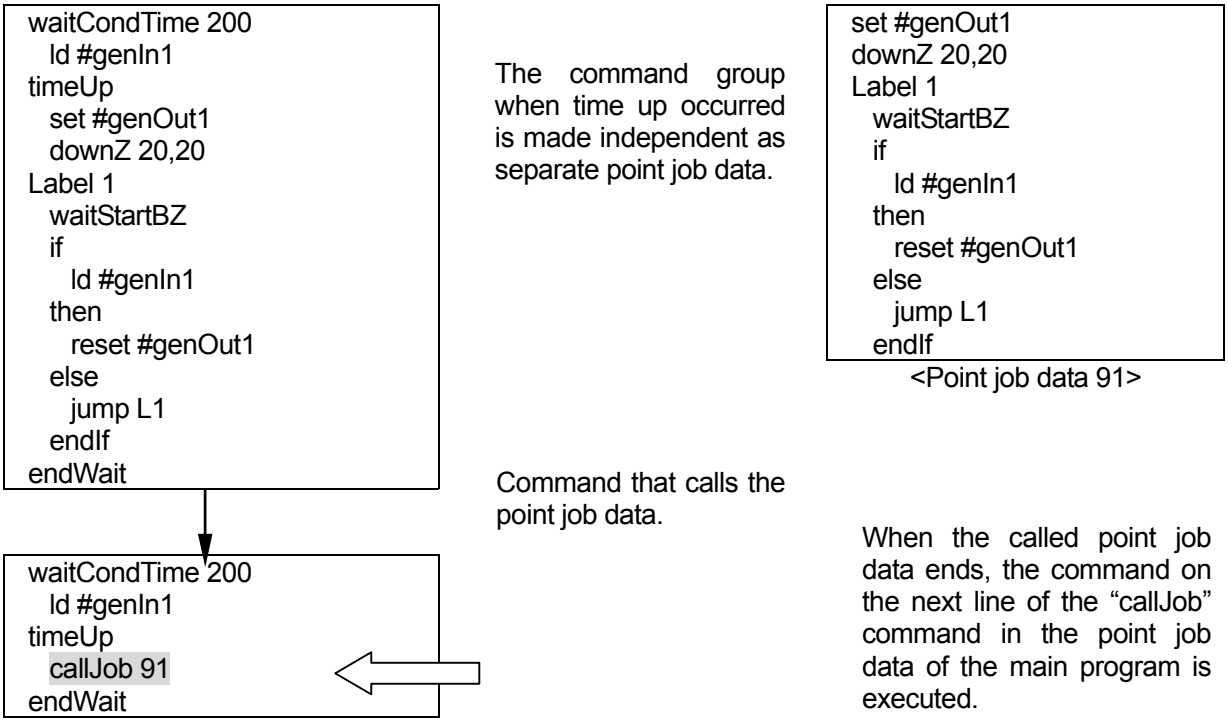
Subroutine call of point job data: callJob

While a point job is running, different point job data can be called and executed.

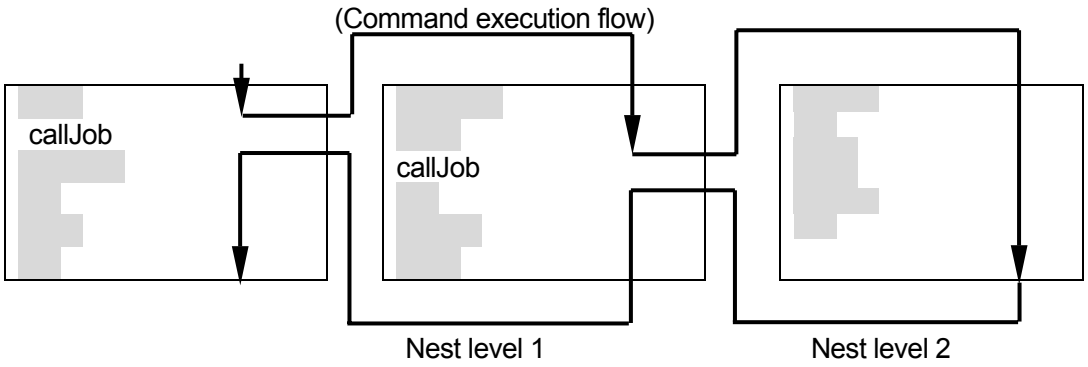
The point job data is reduced and easier to read if error Operation and other parts common to multiple point jobs are made into one point job data and used by calling it from another point job data.
Also, by making a certain command group which was one part of the point job data, one point job data, only that part can be tested.

Command category	Command	Parameters	Job
Execution flow control	callJob	Point job data no.	Calls a subroutine of the point job data for the given number.

- The “callJob” command is invalid at the CP Passing Point or a point whose Base Type is CP Passing Point.



- When the point job data called by “callJob” command contains a “callJob” command, and the nest level exceeds 10, an error (No. 42) is generated. (↓ Example of nest level 2)



Point job data number can also be given by expression.

Example)

```
declear num ejob
waitCondTime 200
  ld #genIn1
timeUp
if
  ld #genIn2
then
  ejob = 9
else
  ejob = 10
endIf
 callJob ejob
endWait
```

Local variable “ejob” declaration
Waits for 0.2 seconds until the following conditions are met.
  #genIn1=ON (condition)
When the condition is not met in 0.2 seconds,
if
  #genIn2=ON
then
Assigns 9 to “ejob”.
If not,
Assigns 10 to “ejob”.

Calls a subroutine of the No. 9/10 point job data.

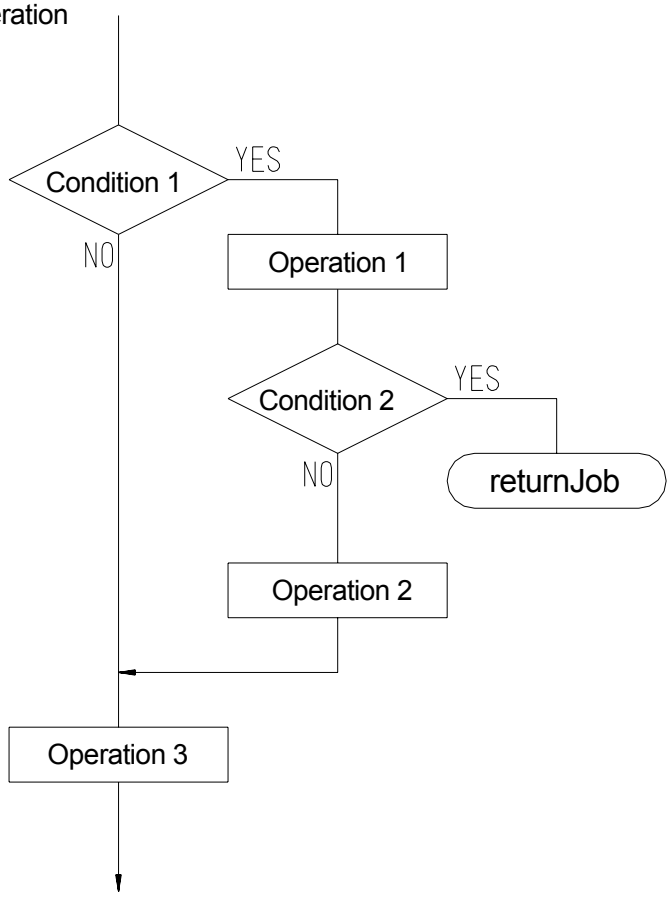
End of point job: returnJob

When there are complex conditions and operations that correspond to them and there are no more operations in the point job, the point job can be ended by “returnJob” command.

Command category	Command	Parameters	Job
Execution flow control	returnJob		Ends a point job.

e.g.) Below is the Point Job Data for the operation shown in the chart to the right:

```
if
  Condition 1
then
  Operation 1
  if
    Condition 2
  then
    returnJob
  else
    Operation 2
  endIf
endIf
Operation 3
```



If ”returnJob” is omitted, ”Operation 3” will be executed even if “Condition 2” is ON (YES.)

Subroutine call of Program: callProg

The following explains how to call and execute other programs while running a point job.

Command Category	Command	Parameter	Job
Execute Flow Control	callProg	Program Number	Call a subroutine of a program specified by number.

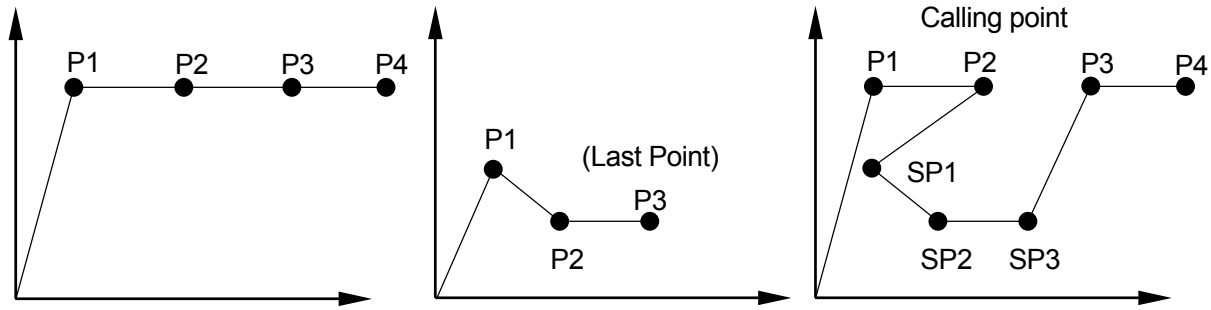
- The command is invalid at the CP Passing Point or a point whose Base Type is CP Passing Point.

```
waitCondTime 200
  ld #genIn1
timeUp
  callProg 17
endWait
```

This is the command to call out Program Number 17.

After executing the called program, it starts to execute the command in the next line of the "callProg" command in the calling program.
("endWait" in this case)

- The called program (subprogram) is run in one cycle regardless of the setting. It does not return to the work home position. Refer to the illustration below. (SP1: Subprogram Point 1)



Also, Program Number can be set using expressions.

Example)

```
declear num eprg
waitCondTime 200
  ld #genIn1
timeUp
if
  ld #genIn2
then
  eprg = 9
else
  eprg = 10
endIf
 callProg eprg
endWait
```

Declare the local variable “eprg.”
Wait for 0.2 sec until the following condition is met.
#genIn1=ON (Condition)
If the conditions are not met within 0.2 sec,
If
#genIn2=ON
then
Assign 9 to “eprg.”
If not,
Assign 10 to “eprg.”

Call a subroutine of Program number 9/10.

Point Data Setting

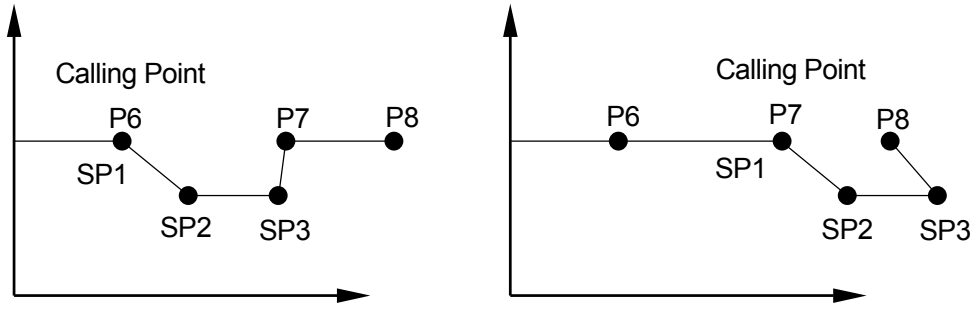
[Position Data] of the program data defines how to handle coordinates (position data) in the point data. It consists of the following three types.

Absolute Coordinates: position data value is deemed as the fixed coordinates of the robot.
Relative Coordinates: position data value is deemed as the distance from coordinates to where the program starts.
Moving Amount: position data value is deemed as the distance to the next point.

If you set the subprogram to [Relative] or [Moving Amount], the tool unit always runs at an equal distance from the called point (to where point job data including “callProg” command is assigned.)

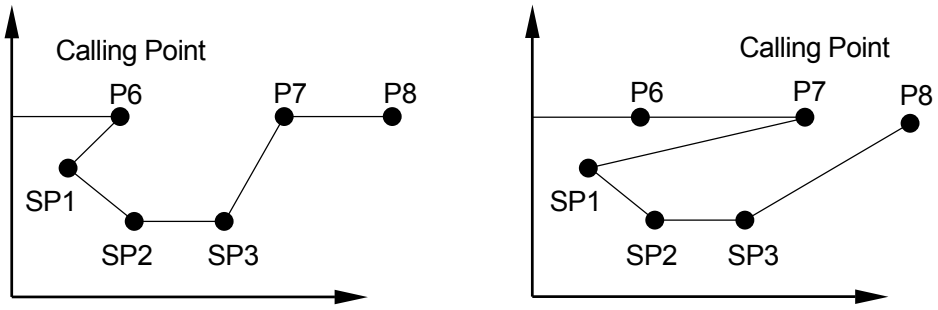
Example: The subprogram is set to [Relative] or [Moving Amount.]

The current point (called point) is P1 (SP1) (excluding P1 coordinate data in the subprogram.) The work home position is ignored.



Example: The subprogram is set to [Absolute.]

The tool unit runs on the coordinates of the point data regardless of the position of the called point. It executes the point job at the start in the work home (in the subprogram) at the current point (called point) and then shifts to P1 (SP1.)



- When the “callProg” command is included in a program that is called by a “callProg” command, an error is returned if the nest level exceeds 10.

The coordinates (position data) included in the point data can be selected from [Absolute], [Relative] and [Moving Amount.] The default value is set to [Absolute.]

Absolute Coordinates: position data value is deemed as the fixed coordinates of the robot.

Relative Coordinates: position data value is deemed as the distance from the coordinates where the program starts. (If the start coordinate is (0,0) it will be the same as [Absolute.])

Moving Amount: position data value is deemed as the distance to the next point.

Depending on handling of the position data, the moving point positions vary even if the values are the same. Refer to the following examples.

Point Data Coordinates		(0,0)	(15,20)	(10,20)	(5,10)	
	P1	P2	P3	P4	P5	
Absolute	(10,10)	(0,0)	(15,20)	(10,20)	(5,10)	Absolute coordinates
Relative	(10,10)	(10,10)	(25,30)	(20,30)	(15,20)	Absolute coordinates
Moving Amount	(10,10)	(10,10)	(25,30)	(35,50)	(40,60)	Absolute coordinates

Program Start Coordinates

- If you run the program as a subprogram, the tool unit will not return to the work home. If the program is set to “Relative” or “Moving Amount”, the tool unit will also not return to the work home.
  The tool unit returns to the work home only when the program is set to “Absolute” and executed independently (not “callProg” running.)

Registering the program set to “Relative”
If you register a point in JOG mode, you have to select “Absolute” regardless of the position data setting.
If you create a “Relative” program, shift all points (offset) so that the coordinates of the first point become (0,0,0) after registering the point.

Registering a program set to “Moving Amount”
The registered coordinates cannot be converted into “Moving Amount.” Register the point in MDImode.

Calling points: callPoints

Call a point string (defined in customizing mode) with identifier to execute it.

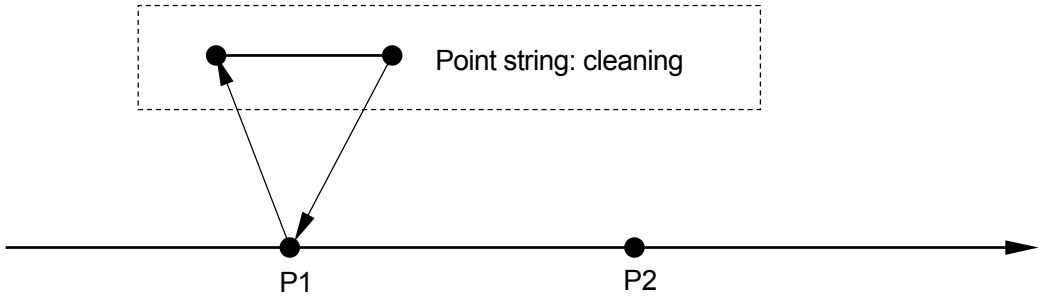
Command Category	Command	Parameter	Job
Execute Flow Control	callPoints	Point String Identifier	Calls a subroutine of the specified point string.

- The command “callPoints” is invalid at the CP Passing Point or a point whose Base Type is CP Passing Point.

Example) The following point job data is set to P1.

```
if
   ld #genIn1
then
   callPoints cleaning
downZ 10,10
out #genOut1
```

Command used to call a point string “cleaning”



If #genIn1 is ON, go to the point string “cleaning” and execute point job data and additional function data set to “cleaning”. Then go to P1, lower the Z axis by 10 mm and output ON signal to #genOut1.
If #genIn1 is OFF, lower the Z axis by 10 mm and output ON signal to #genOut1.

Ending a program: endProg

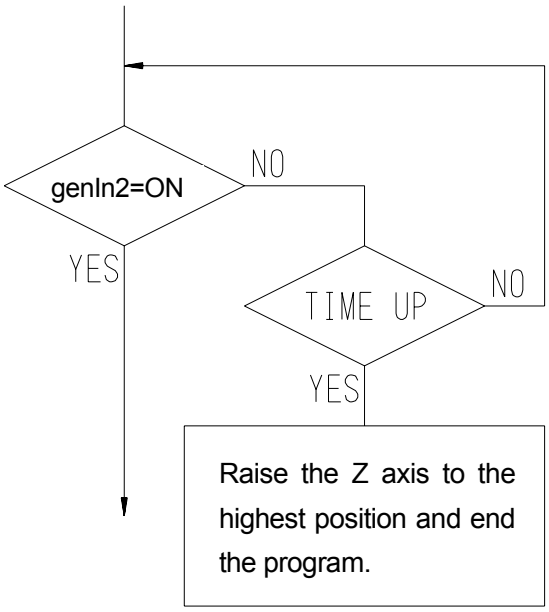
The following explains how to end a program (running the operation) at the current point. The arm will not return to the Work Home position.

Command Category	Command	Parameter	Job
Execute Flow Control	endProg	-	Ends program run at the current point.

- The command “endProg” is invalid at the CP Passing Point or a point whose Base Type is CP Passing Point.

Example: Below is the The Point Job Data for the operation shown to the right chart

```
waitCondTime 500
  Id #genIn2
timeUp
  movetoZ 0,10
  endProg
endWait
```



“endProg” is a command to end the program at the current point without returning to the Work Home position.
It is different from stop in that the robot will not restart operation. You have to start the job from the beginning.

- If you want the robot to return to the Work Home position before ending the program, use the “goPoint” command with a destination number “0” for the Work Home Position.

Assigning the returned value of a function: returnFunc

Assign a value of the specified expression as a returned value and end the function.

Command Category	Command	Parameter	Job
Execute Flow Control	returnFunc	Expression	Assigns a value of the specified expression as a returned value and end the function.

- The command “returnFunc” cannot be used for point job data.

Jopint Job Data

```
outLCD 7,4,radians(x)
```

Call a function.

Function (Identifier: radians)

```
returnFunc 0.017453*x
```

A returned value of “radians” function for Argument(x) is displayed on the teaching pendant LCD.

Jumping to a specified point: goPoint, goRPoint, goCRPoint

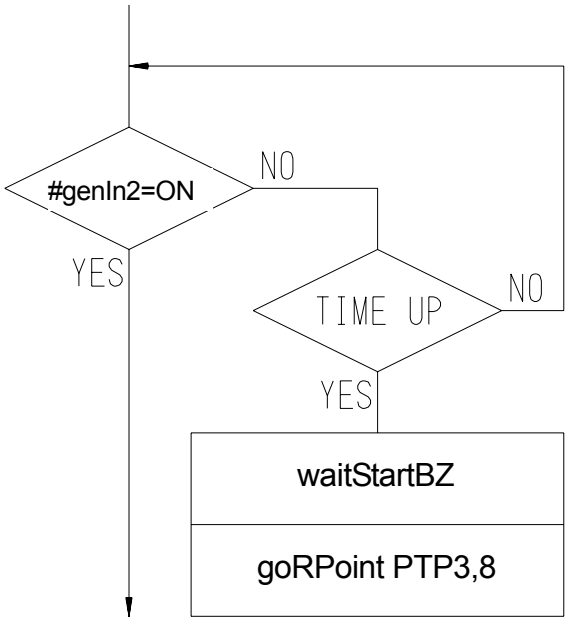
The following explains how to jump to a specified point after carring out a point job instead of going to the next point.

Command Category	Command	Parameter	Job
Execute Flow Control	goPoint	PTP Condition Number, Point Number	Jumps to a specified point.
	goRPoint	PTP Condition Number, Relative Point Number	Jumps to a specified relative point.
	goCRPoint	PTP Condition Number, Destination selection	Jumps to a specified destination while running in CP drive.

- The commands “goPoint”, “goRPoint” and “goCRPoint” are invalid at the CP Passing Point or a point whose Base Type is CP Passing Point.
- Point Number or Relative Point Number of “goPoint” and “goRPoint” can be set using variables or expressions.
  A destination for “goCRPoint” can also be set using variables or expressions. In this case, the value must be either “0” or “1,”

Example: Below: is the Point Job Data for the operation shown in the chart to the right.

```
waitCondTime 500
  ld #genIn2
timeUp
  waitStartBZ
  goRPoint PTP3,8
endWait
```



The above commands carry out the following operation:
When #genIn2 does not turn ON within 0.5 sec, the alarm buzzer will sound and the robot will wait until a start signal is received. After a start signal is received, operation will restart at the current reference point, plus 8.

[goPoint PTP3,25]: Jump to Point 25. (Comply with PTP Condition 03.)

If you set “0” as the PTP Condition Number, the movement will comply with the PTP Conditions setting in the program data.

If you set “o” as the point number, the arm will go to the Work Home position.

(Jump to a point specified by number.)

[goRPoint PTP3,-4 : Subtract four from the current reference point number and jump to the point with that number. (Comply with PTP Condition 03.)

If you set “0” as the PTP Condition Number, the movement will comply with the PTP Conditions setting in the program data.

If you set “0” as the Relative point number, the robot will restart at the current point.

(Jump to a relative point specified by number.)

[goCRPoint PTP3,1 : This command is used to jump to a specified point while running in CP drive.

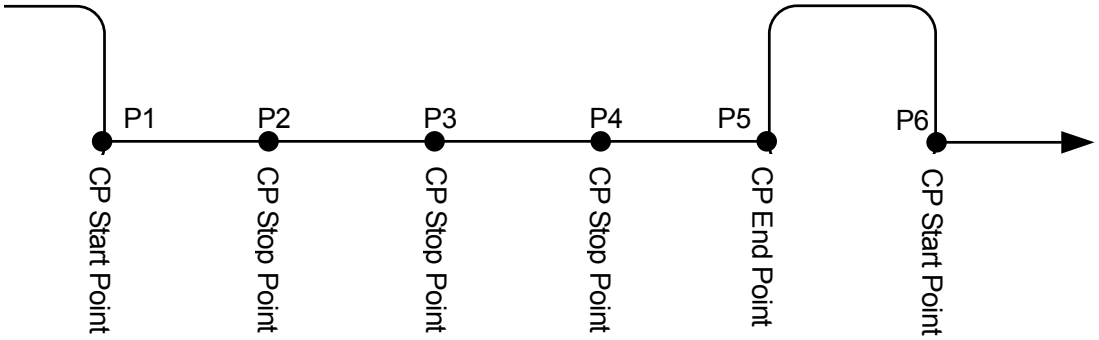
The movement from a CP Start Point to a CP End Point is regarded as one action. If you set 0 as the destination, the arm will return to the point where it started the operation in CP drive (CP Start Point.) (PTP Condition is 03.) If you set 1 as the destination, the arm will jump to the next point of a CP End Point. (PTP Condition is 03.)

If you set 0 as the PTP Condidtion Number, the robot will comply with the PTP Condition setting in the program data.

Example: If this command is executed between P1 and P5, the arm will shift as shown below

Destination 0: Shift to P1.

Destination 1: Shift to P6.



Jumping to a specified command line: jump, Label

Command Category	Command	Parameter	Job
Execute Flow Control	jump	Label Number	Jumps to a “Label” specified by number.
	Label	Label Number	Destination mark to “jump” to.

Example: If #genIn2 is ON, the alarm buzzer sounds and stands by until a start signal comes.
If #genIn2 is not ON, go to the next job.

Label 1	(Destination mark)
if	If the following condition is true, go to then. If not, go to the next of endIf.
ld #genIn2	genIn2=ON
then	If the condition is true, execute the following commands.
waitStartBZ	Sound a buzzer and stand by in place until a start signal is received.
jump L1	Jump to [Label 1.] (when a start signal is received.)
endIf	End of if Branch

- The command “Label” cannot be set between “if” and “endIf”, or “waitCond” and “endWait.”
- Label can be set from “Label 1” up to “Label 99.”

[For, Do-loop]

For, Do-loop: for, next, exitFor, do, loop, exitDo

Command Category	Command	Parameter	Job
for, do-loop	for	Control Variable, Initial Value, End Value, Step Value	Repeats commands from “for” to “next” until the specified variable changes from Initial Value to End value.
	next	-	
	exitFor	-	Exits from “for” sentence.
	do	-	Repeats commands from “do” to “loop.”
	loop	-	
	exitDo	-	Exits from “do” sentence.

for ~ exitFor ~ next

“for” is a command to specify the number of repetitions.

```
declear num ival
for ival=1 to 8 step 1
  (contents of repetition)
next
```

Declare the local variable “ival.”
The initial value of the variable “ival” is 1. Add to the variable by one for every looping and repeat the commands from “for” to “next” until “ival” becomes 8.

```
declear num ival
for ival=1 to 8 step 1
  (contents of repetition)
  if
    ld #genIn1
  then
    exitFor
  endIf
next
```

Declare the local variable “ival.”
“exitFor” is a command to exit from the repetition of “for next” and go to the next command of “next.”

Condition: If #genIn1=1, exit from the repetiion of “ next” even if “ival” does not become 8 and go to the next command of “next.”

Parameters (the initial value, end value and step value) of the command “for” can be set using variables or expressions.

```
declear num loop
declear num ival
if
  ld #genIn1
then
  loop = 5
else
  loop = 10
endIf
for ival=1 to loop step 1
  (contents of repetition)
next
```

Declare the local variable “loop.”
Declare the local variable “ival.”
If
  #genIn1=ON
then
  Assign 5 to “loop.”
If not
  Assign 10 to “loop.”

The initial value of the variable “ival” is 1. Add to the variable by one for every looping and repeat “for next” until the value of the “ival” become the same as the variable “loop (5/6).”

do ~ exitDo ~ loop

Repeat an operation from “do” to “loop” until it exits by “exitDo.”

```
do
  (contents of repetition)
loop
```

If there is no condition to exit from repetition, repeat looping endlessly.

```
do
  (contents of repetition)
  if
    ld #genIn1
  then
    exitDo
  endIf
  (contents of repetition)
loop
```

- (contents of repetition) can be put before or after Condition.

Condition:
If #genIn1=1, exit from the repetition of “do ~ loop” and go to the next command of “loop.”

- In case of the looping commands, an error will occur if the nest level exceeds 10.
- If you set the looping commands at CP Passing Point or a point whose Base Type is CP Passing Point as a point job, the robot may stop because of too many loops.

[Controlling Tool Movement]

Moving the Z axis: upZ, downZ, movetoZ

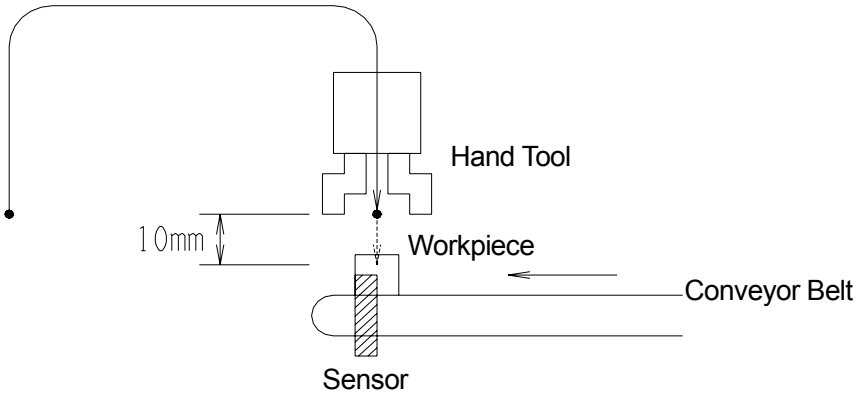
The following explains how to raise/lower the Z axis only by setting a point job. These commands belong to the category [Move.]

Command Category	Command	Parameter	Job
Move	upZ	Distance, Speed	Raises only the Z axis by the specified distance.
	downZ	Distance, Speed	Lowers only the Z axis by the specified distance.
	movetoZ	Distance, Speed	Raises or lower the Z axis to the specified Z coordinates (Absolute coordinates.)

- Move commands are invalid at the CP passing point and points whose Base Type is CP Passing Point the base type.

Example:

1. The current movement condition is PTP movement.
2. The arm stops descending before holding the workpiece.
3. The sensor detects the workpiece.
4. The hand tool descends slowly to hold the workpiece.




```
waitCond
  ld #genIn2
endWait
downZ 10,20
```

Wait in place until the following conditions are met.
#genIn2=ON (Condition)
End of conditions.
Descend only the Z axis at the speed of 20 mm/sec.

The distance or speed can be set using variable or expressions.

```
waitCond
  ld #genIn2
endWait
downZ #P_Z(1)-#point_Z,20
```

Wait in place until the following conditions are met.
#genIn2=ON (Condition)
End of conditions
Lower or raise only the Z axis at the speed of 20 mm/sec by a distance calculated by deducting the Z coordinates of the current point from the Z coordinates of P1.

#P_Z(1): Variable which has the Z coordinates of P1 as a value in the current program.
#point_Z: Variable which has the Z coordinates of the current point as a value.

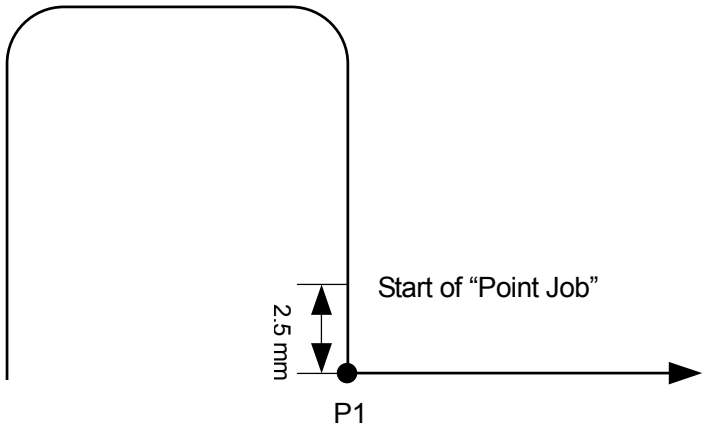
- If you assign a value to the variable "#jobStartHight" for a "Job before Moving" or a "Job while Moving" (let), the robot starts a point job from an assigned value above the set point Z coordinate.

Example)

P01
Type: CP Start Point
Job before Moving: Point Job Data 3
Point Job: Point Job Data 12

Point Job Data 3

```
#jobStartHight 25
```



Moving straight in CP drive:

lineMoveSpeed, lineMoveStopIf

The following explains how to move straight in CP drive using point job data commands.
The speed of CP drive and the moving amount of each axis coordinates can be set. You also can end shifting by setting conditions.

Command Category	Command	Parameter	Job
Move	lineMoveSpeed	Speed (CP speed) X Distance Y Distance Z Distance R Rotate Angle	Moves by the entered distance in CP drive.

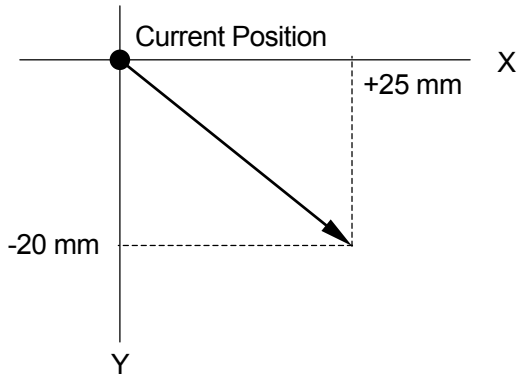
- Move commands are invalid at the CP Passing Point or a point whose Base Type is CP Passing Point.

Enter not the coordinates but the distance from the current point to the destination point where you want to shift as Distance. Enter "0" for the direction in which you do not want to shift.
The distance can be entered using a variable or an expression instead of a value.

Shown to the right are the parameters of these command, lines from "lineMoveSpeed" to "endLineMove".

```
lineMoveSpeed 20
  X:25
  Y:-20
  Z:5
  R:0
endLineMove
```

Each command is displayed for each coordinate axis, but the movements of the axes are executed at once.



Stopping the arm shift due to setting conditions while running.

```
lineMoveSpeed 3
  X:20
  Y:0
  Z:0
  R:0
lineMoveStopIf
  ld #sysIn1
endLineMove
callJob11
```

Condition to stop shifting

If #sysIn1 comes ON, the tool unit stops shifting and goes the next command (callJob11) even before the movement in the Z direction does not reach +20.

In this case, you can check with the system flag (#sysFlag34) to find out whether it stopped before or after shifting to the specified distance was complete.

0: Shifting to the specified distance is complete.

1: Shifting to the specified distance is not complete due to conditions.

Exceeding the “Move Area Limit” of “lineMoveSpeed”

If the arm exceeds the move area limit, it will stop shifting at the position where it reaches the move area limit and advance the next command.

In this case, you can check with the system flag (#sysFlag33) to find out whether it stopped before or after shifting to the specified distance was complete.

Complete: 0, Not complete: 1

If the arm achieves the move area limit before the shifting is complete, sound a buzzer and stop. (Wait Start)

```
lineMoveSpeed 3
  X:20
  Y:0
  Z:0
  R:0
endLineMove
if
  ld #sysFlag33
then
  waitStartBZ
endIf
```

Executing mechanical initialization by a point job: initMec

- The following commands are valid for the CARTESIAN Series only.

The following explains how to execute mechanical initialization (executed when the power of the robot is turned ON) using point job commands. Even if a position error occurs, the tool unit returns to the absolute coordinates (x:0, y:0, z:0, r:0) by executing mechanical initialization.

Command Category	Command	Parameter	Job
Move	initMec	Axis specification	Execute mechanical initialization for a specified axis.

- Move command is invalid at the CP Passing Point or a point whose Base Type is CP Passing Point.

Axis specification	Contents
All	Execute mechanical initialization for all the axes.
x	Execute mechanical initialization for the x axis.
y	Execute mechanical initialization for the y axis.
z	Execute mechanical initialization for the z axis.
r	Execute mechanical initialization for the r axis.

- Mechanical initialization is executed at low speed.

Position error detection: checkPos

- The following commands are valid for the CARTESIAN Series only.

The following explains how to detect a position error using a point job command.
If the checkPos command is executed, the tool unit goes to the absolute coordinates (x:0, y:0, z:0, r:0) regardless of where the tool unit is. After position error detection, it goes to the next point.

Command Category	Command	Parameter	Job
Move	checkPos		Detects position errors.

- The Move command is invalid at the CP Passing Point or a point whose Base Type is a CP Passing Point.

For the results of position error detection, refer to the system flag (#sysFlag(35).)
Normal: 0, Position Error: 1

Example)

```
checkPos
if
  ld #sysFlag(35)
then
  waitStartBZ
endIf
```

If position error detection is executed and a position error is found, a buzzer sounds and the robot waits for start.

If a position error is detected, the #sysOut8 (position error) signal also comes ON.

[LCD, 7SLED]

Displaying the specified strings on the teaching pendant:

clrLCD, clrLineLCD, outLCD, eoutLCD

The following explains how to display/ not display entered items on the teaching pendant LCD.

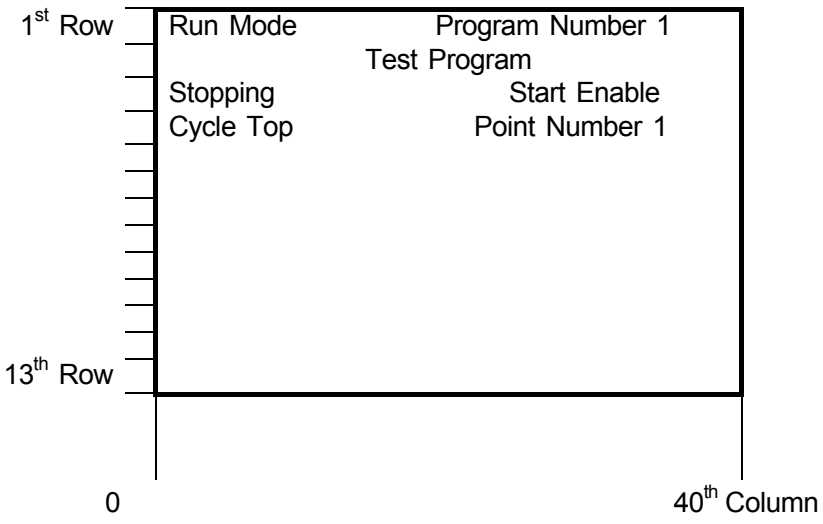
Command Category	Command	Parameter	Job
LCD Control	clrLCD	None	Clears the LCD display.
	clrLineLCD	Rows	Clears only the specified line on the LCD display.
	outLCD	Row, Column, Character String	Displays strings or string variables enterd at a specified position on the LCD display.
	eoutLCD	Row, Column, Character String Expression	Displays the result of a character string expression entered a specified position on the LCD display.

- Rows or columns can be specified using variables or expressions. Also, a string can be specified using string expressions for the command “outLCD.”

  outLCD 7,4,"PULSE": Display the string “PULSE” on the teaching pendant LCD.

  outLCD 7,4,#sv(24) + #sv(25): Display the combined value of the string expressions #sv(24) and #sv(25) on the teaching pendant LCD.

There are 13 rows and 40 columns on the teaching pendant LCD.
A maximum of 40 characters in single byte, 20 characters in double byte can be displayed in one line.



Displaying arbitrary numbers on the 7SLED: sys7SLED, out7SLED

- The following commands are valid for the CARTESIAN Series only.

The following explains how to display arbitrary numbers on the 7SLED on the front body using "out7SLED" command.
When executing "sys7SLED" command or switching programs, the program number will be displayed again.

Command Category	Command	Parameter	Job
LCD, 7SLED	sys7SLED	-	Display the previous program number before changed by "out7SLED" command.
	out7SLED	Display Type Display value	Outputs number to 7SLED.

Parameter display type of "out7SLED" can be selected from the following 4 items. Display value can also be specified using variables or expressions.

Display Type	Description
Num	Displays a specified number on the 7 SLED.
Er	Displays one after the other between "Er" and a specified number.
St	Displays one after the other between "St" and a specified number.
Ur	Displays one after the other between "Ur" and a specified number.

Example: out7SLED Num,10: Display numeric value "10."
out7SLED Er,20: Display one after the other between "Er" and "20."
out7SLED St,nMyNum: Display "St" and a value of the variable "nMyNum."

[COM Input/Output]

COM Input/Output: outCOM, eoutCOM, setCOM, cmpCOM, ecmpCOM, clrCOM, shiftCOM

Data can be output or input from the COM.

Command Category	Command	Parameter	Job
COM Input/Output	outCOM	Port, Character String	Outputs a character string from the COM.
	eoutCOM	Port, Character String Expression	Outputs the result of an expression from the COM.
	inCOM	Variable Name, Port, Wait Time	Assigns data received from the COM to the specified variable.
	setWTCOM	Port, Wait Time	Sets Wait Time (Period for Time Out) for receiving from the COM.
	cmpCOM	Port, Character String	Compares received data with a character string. The result is entered into System Flag (sysFlag(1) to (20).)
	ecmpCOM	Port, Character String Expression	Compares received data with a character string expression. The result is entered into System Flag (sysFlag(1) to (20).)
	clrCOM	Port	Clears a buffer received from the COM.
	shiftCOM	Port, Shift Number	Shifts data received from the COM. Deletes data from the top to the Shift Number.

COM Output: outCOM, eoutCOM

A character string with up to 255 characters can be output from the COM.

Select a port (COM No.) under [outCOM]/[eoutCOM] to display the Character Entry screen. Select the desired character string and press ESC. (For key operation on Character Entry screen, see "Character Entry".)

For [eoutCOM] command, characters can be specified in hexadecimal code by using "%." If you want to display "%" on the screen, enter "%%."

e.g.: eoutCOM port2,"%0D%0A" : Outputs CR • LF code.
eoutCOM port2,"%%300" : Outputs %300.
eoutCOM port2,#sv(24) & #sv(25) : Outputs the value combined character string variables #sv(24) and #sv(25.)

- If there is any character other than 0 to 9, A to F or % after "%", "%" is dealt with as a character.

COM Input: inCOM

A specified number of characters out of data received from the COM is assigned to a variable. If received data exceeds the specified number of characters, characters counted from the top by the specified number are assigned.

If received data is less than the specified number of characters, the robot stands by for a time specified in [setWTCOM] and data which has been received is assigned to a variable. If [setWTCOM] is not used, the tool unit stands by for 0.1 sec.

- If point job data including the COM Input command is set at a CP Passing Point, the robot stands by for 0 sec to receive data.

Comparison of Data received from the COM: cmpCOM, ecmpCOM

Compares COM receive buffer (which is a place where received data is stored) and a specified character string one by one from the top character. The comparison results will be reflected by a system flag.

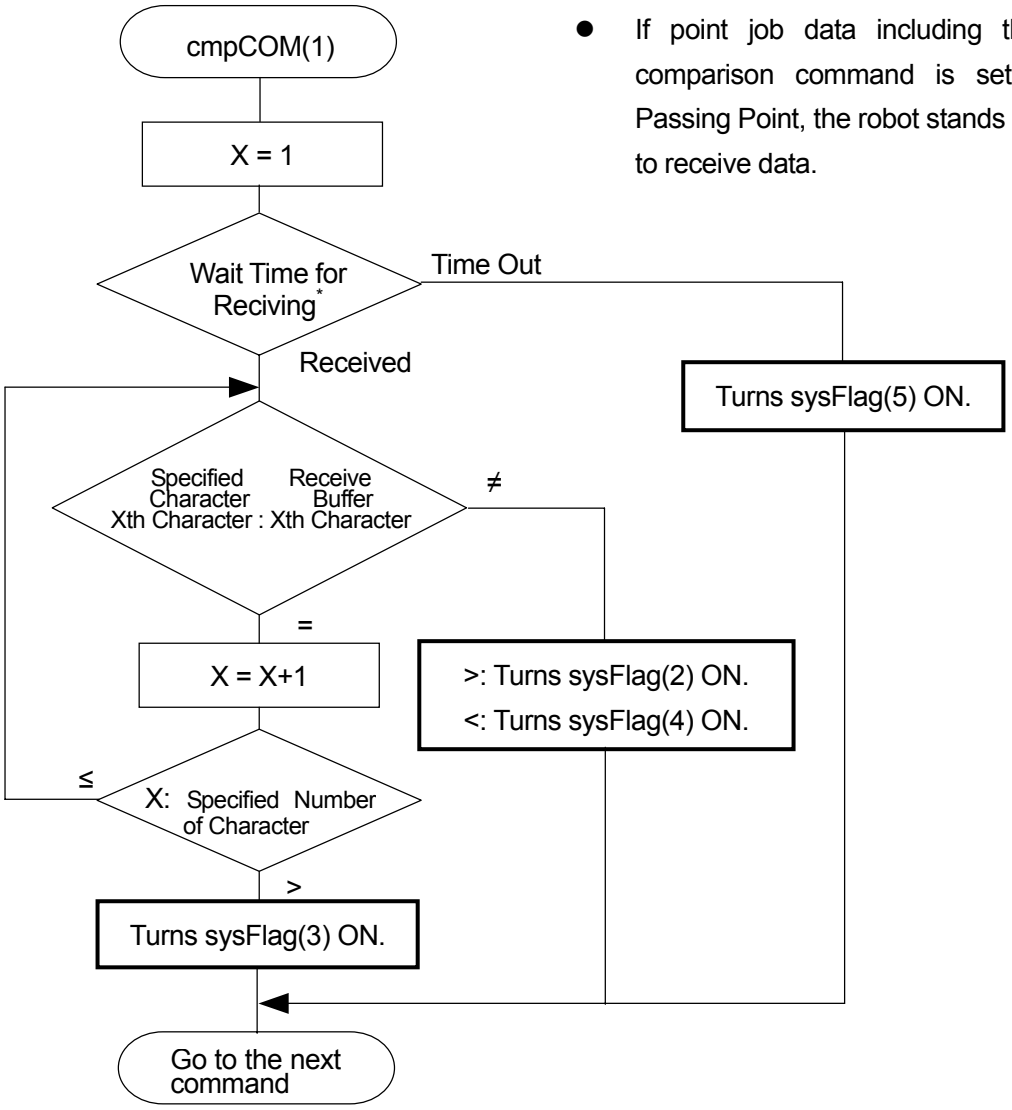
A system flag indicates the condition. e.g. not equal, comparing a specified number of characters is finished or data has not been received after the robot stands by for the set wait time.

You can set a wait time for receiving data using [setWTCOM.] If [setWTCOM] is not used, the robot stands by for 0.1 sec as a wait time.

When using [ecmpCOM] command, character strings compared to a receive buffer can be specified using a character string expression.

System Flag

	COM1	COM2	COM3	COM4 (TPU)
Specified Character > Receive Buffer	sysFiag(2)	sysFiag(7)	sysFiag(12)	sysFiag(17)
Specified Character = Receive Buffer	sysFiag(3)	sysFiag(8)	sysFiag(13)	sysFiag(18)
Specified Character < Receive Buffer	sysFiag(4)	sysFiag(9)	sysFiag(14)	sysFiag(19)
Time Out	sysFiag(5)	sysFiag(10)	sysFiag(15)	sysFiag(20)



- If point job data including this receive comparison command is set at a CP Passing Point, the robot stands by for 0 sec to receive data.

Setting Wait Time for receiving data from the COM: setWTCOM

You can set Wait Time for receiving data using “inCOM” or “cmpCOM” command. If no data is received after a specified time goes by, it will be recognized as Time out. (Turns a system flag ON.)
0.1 sec is set for the defalt wait time.

Clearing COM Receive Buffer: clrCOM

Receive buffer is a place where received data is stored. Each port has a receive buffer (8 kbyte.)
New received data will not replace the existing receive data but will be written after the existing data.
A receive buffer will be cleared by turning the power OFF or executing “clrCOM” command.

Shifting COM Receive Data: shiftCOM

A specified data byte of the receive buffer is deleted.

e.g.: 2-byte Shift

Receive Buffer

- You can tell whether the data is stored in each receive buffer by its system flag.
  If a receive buffer has received data, a corresponding system flag comes ON.

	COM1	COM2	COM3	COM4 (TPU)
With received data	sysFiag(1)	sysFiag(6)	sysFiag(11)	sysFiag(16)

PC Communication: stopPC, startPC

Stopping/Starting COM1 communication stopPC, startPC

COM1 is normally used to communicate with the PC. If you use COM1 not to communicate with PC (for sending/receiving C&T data) but to connect to devices to control the robot by point job commands, it is necessary to stop PC communication transaction operated by the system.

If stopPC command is executed, PC communication does not work until the power is turned OFF or startPC command is executed.

- After executing “stopPC” command, C&T data cannot be sent/received. Please preferably use a connector other than COM1 to connect devices.

[Variables, Comments, System Control]

Declaration and assignment of variable: declear, let

A variable that is valid only in point job data containing a declaration command and a user function (customizing mode) is known as a “local variable”.
A local variable sets the type and identifier at declaration time. The identifier is the name of the variable and type can be selected from either “numeric type” or “string type”.
Also, a local variable can be declared as an array of up to 3 dimensions.

The “let” command assigns a right side numeric value, the value of a variable, or the result of calculation of an expression to the left side variable. When this command is input, only an expression is displayed.

Command category	Command	Set parameters	Job
Variable, comment, system control	declear	Type, identifier	Local variable declaration
	let	Assignment expression	Assigns the result of calculation of the right side expression to the left side variable

e.g.) declear command

```
declear numeric abc
declear string def
```

Numeric variable “abc” declaration
String variable “def” declaration

e.g.) let command

```
count = 0
count = count + 1
count = in - out

total = nin * 365

tsuki = total / 12

fullname=name1 & name2
```

Assigns 0 to variable “count”
Loads 1 in variable “count”
Assigns the difference of the out value subtracted from the in value into variable “count”
Assigns the product of 365 multiplied by the value of nin to variable “total”
Assigns the quotient of the value of total divided by 12 to variable “tsuki”
Assigns the string connecting name1 and name2 to variable “fullname”

Both of the following point job data use the local variable “count”, but since a local variable is a function which is valid only in point job data containing a declaration command, they do not interfere. For example, 0 is assigned to “count” at point job data 24, but the value of “count” used by point job data 05 does not change. The opposite also applies.

e.g.) Joint job data 05

```
declear numeric count
count=0
do
  count=count+1
  callJob 24
  if
    ld count>=10
  then
    exitDo
  endIf
loop
```

“count” local variable declaration (numeric type)
Sets initial value 0 into “count.”
Repeats up to the loop command.
  Loads 1 into “count”.
  Executes point job data 24.
  If
   the value of “count” was 10 or greater,

Jumps to the command after the loop command.

Returns to the do command.

e.g.) Point job data 24

```
declear count
count=0
 Label 1
pulse #genOut11,250
count=count+1
if
  ld count<=3
then
  jump L1
endIf
```

“count” local variable declaration
Sets initial value 0 into “count”.
Label 1 (jump destination mark)
ON pulse output to #genOut1”.
Loads 1 into “count”.
If
  the value of “count” was 3 or less,

  jumps to label 1.

Comment insertion: rem,crem

Comments can be added to point job data and sequencer program commands.

Command category	Command	parameters	Job
Variable, comment, system control	rem	String	1 line comment
	crem	String	Comment in the end of a command line

e.g.)

```
if
  ld #genIn1
  rem #genIn1 Obstruction sensor
then
  waitStartBZ
```

If
#genIn1 is true,
(#genIn1: Obstruction sensor) : Comment line

Sounds a buzzer and stands by until start is given.

e.g.)

```
if
  ld #genIn1 crem #genIn1 Obstruction sensor
then
  waitStartBZ
```

If
#genIn1 is true,
(#genIn1: Obstruction sensor) : Comment line

Sounds a buzzer and stands by until start is given.

- For a teaching pendant, when the comment is longer than 1 line even when “crem (end of line comment)” is used, line feed is performed to display the comment.

Changing a program number using point job: setProgNo

The program number being selected can be changed by the point job. It is available in the following cases.

- If you set a program number by the point job after the power is turned ON, the same program number will always be activated when the power is turned ON.
- If you set a program number by the point job at the end at the work home, you can switch it to the next program number. This fuction is useful for executing a series of programs in order. For example, a series of point jobs such as [Program 1 Program 2 Program 3] is executed repeatedly. If [SET ProgNum2] is executed by the point job at the end at the work home in Program 1, the program changes to Program 2 after running Program 1. You can also set the programs to change from 2 to 3 and to 1 repeatedly.
- You can set a program number according to input from COM using the point job at the work home. You can connect a barcode reader to COM and change the program according to the value of the barcode.

If you change the number while running a program, the running program or sequencer program will not be changed instantly. After changing the number, the running program will be changed when the robot restarts from the standby state.
If you want to execute another program while running the program, use the [callProg] command.

Command Category	Command	Parameter	Job
Variable, Comment, System Control	setProgNo	Program Number	Changes the program number when the robot starts running after Wait Start.

- The program number can be set using variables or expressions.

Changing a sequencer program using point job: setSeqNo

The sequencer number being selected can be changed using the point job.
However, a complicated command cannot be created because the number of commands for sequencer program is 100 steps maximum. Therefore, you need to create some sequencer programs (executed when the power is turned ON, during standby and during operation) separately so that you can switch between the programs using “setSeqNo.”
For example, if you set “setSeqNo02” for “Job on Start of Cycle (Job on Run Mode)” and “setSeqNo01” for “Job on End of Cycle (Job on Run Mode)”, the 2nd sequencer program will be executed during the operation and the 1st sequencer program will be executed during standby.

If you change the number while running a program, the running program or sequencer program will not be changed instantly. After changing the number, the running program will be changed when the robot restarts from standby state.

Command Category	Command	Parameter	Job
Variable, Comment, System Control	setSeqNo	Sequencer Program Number	Changes the sequencer program number when the robot starts running after Wait Start.

- The sequencer program number can be set using variables or expressions.

Warranty

Henkel Corporation warrants, to the original Buyer for a period of one (1) year from date of delivery, that the Loctite® Equipment or System sold by it is free from defects in material and workmanship. Henkel will, at its option, replace or repair said defective parts. This warranty is subject to the following exceptions and limitations.

1. Purchaser Responsibilities – The Purchaser shall be responsible for:
   -Maintenance of the equipment as outlined in the Equipment Manual for the product.
   - Inventory of recommended maintenance parts established by Henkel;
   -Notification to Henkel within 6-8 hours of downtime.
   -Any cost of travel or transportation connected with warranty repair.
   -All cost associated with investigating or correcting any failure caused by the purchaser's misuse, neglect or unauthorized alteration or repair.
   -All costs attributed to accident or other factors beyond Henkel's control.

2. A thirty (30) day warranty will be extended on any items subject to normal wear, such as:
   -Pump Seals -Tubing -Wear Surfaces of Wiping Rollers
   -O-Rings -Hoses

Purchased items used in Loctite® dispensing equipment are covered under warranties of their respective manufacturers and are excluded from coverage under this warranty. Typical purchased items are:

-Solenoids -Electrical Relays -Refrigeration Units
-Timers -Fluid Power Cylinders -Electrical Motors

3. No warranty is extended to perishable items, such as:
   -Fuses -Dispensing Needles -Dispensing Nozzles
   -Light Bulbs -Lamps -Product Barrels

Henkel reserves the right to make changes in design and/or improvements to its equipment without obligation to include these changes in any equipment previously manufactured.

Henkel's warranty herein is in lieu of and excludes all other warranties of Henkel and its affiliated and related companies (hereinafter the "seller companies"), express, implied, statutory, or otherwise created under applicable law including, but not limited to, any warranty or merchantability and/or fitness for a particular purpose of use. In no event shall the seller and/or the seller companies be liable for any direct, indirect, special, incidental or consequential damages, including, but not limited to, loss of profits. In addition, this warranty shall not apply to any products, which have been subjected to abuse, misuse, improper installation, improper maintenance or operation, electrical failure or abnormal conditions; and to products, which have been tampered with, altered, modified, repaired or reworked by anyone not approved by seller. Buyer's sole and exclusive remedy under this warranty shall be limited to, at seller's discretion, the replacement or repair of any defective product or part thereof, or a refund of the purchase price paid by for the product in exchange for buyer's return of the product to seller, free and clear of any and all liens and encumbrances of any nature.

The specifications of the robot or the contents of this manual may be modified without prior notice to improve its quality.

No part of this manual may be reproduced in any form, including photocopying, reprinting, or translation to another language, without the prior written consent of LOCTITE®.

P/N 989524 Features II Manual, 07/2005